STUDIES ON FUZZY COMBINATORIAL OPTIMIZATION

Minoru TADA

Submitted in partial fulfillment of the
requirement for the degree of
DOCTOR OF ENGINEERING
(Applied Mathematics and Physics)

at

Kyoto University
Kyoto, Japan
July, 1994

Preface

The theory of fuzzy sets is used in order to deal with uncertainties of the phenomena in the real world. It plays a vital role when we formalize and analyze the models in which imprecise and ambiguous factors are involved, in particular, for those which are accompanied by human cognitive process. In effect, the fuzzy set theory is a body of concepts and techniques that give a form of mathematical precision to human cognitive elements. Today, these concepts are being rapidly accepted among engineers, scientists, mathematicians, linguists and philosophers. In paticular, fuzzy set theory has been applied to control engineering, artificial intelligence, expert system, management science, pattern recognition, social science and so forth. In the field of Operations Research, the applications of fuzzy set theory have been investigated particularly in relation to linear programming. It has also been frequently applied to the area of statistics, e.g., the so-called statistical data analysis such as linear regression analysis, statistical inference (estimation), clustering and so on.

As far as we know, however, there have been few applications of fuzzy set theory to combinatorial optimization problems. It appears therefore important to generalize the formulations of combinatorial optimization problems and analyze their mathematical properties in the light of fuzzy set theory. The main purpose of this dissertation is to lay a foundation of "fuzzy combinatorial optimization". That is, we try to generalize the combinatorial optimization

problems such as scheduling problems and network flow problems by introducing fuzzy set theory. The "fuzzification" of combinatorial optimization is needed for the following reasons, i.e., (1) the standard solution methods may not fit to actual situations because of the existence of uncertain elements and non-rigid factors involved therein, e.g., human cognitive process, and (2) most of the combinatorial optimization probelms do not have efficient algorithms and it may be possible to ease the situation by relaxing the restrictions in the sense of fuzzy set theory. These two descriptions represents the two sides of the problems, required by a decision maker who faces the problems: (1) "realistic" modeling and (2) "more efficient" solution algorithm. Although it is best if both are achieved simultaneously, these two sides usually conflict each other, and we have to choose one of them; we shall restrict our attention to the former in this dissertation.

The types of "fuzzification" introduced in this dissertation are roughly classified into two, i.e., *Type 1: degree of satisfaction*-type and *Type 2: uncertain data*-type. Type 1 comes from the idea of fuzzy objective function. In other words, we generalize the problems by taking into account the differences of decision makers' subjectivity for a derived solution. For instance, given a certain maximization (minimization) problem, a decision maker may be satisfied with a presented solution even if its objective value is less (greater) than the "exact" optimal one, while another decision maker may stick to the exact optimum value. The former decision maker just wants to realize an objective value that is contained in his/her acceptance region of "subjective optimal values". Hence he/she treats the problem as that of maximizing (minimizing) the degree of satisfaction characterized by the idea of fuzzy objective function. On the other hand, in order to deal with type 2 characteristic, fuzzy sets are used to represent non-rigid (uncertain or ambiguous) data in combinatorial prob-

lems. That is, the combinatorial optimization problems can not be described exactly if the data used to describe the problems are insufficient or behave unsteadily. In this case, we express the data by means of "fuzzy number", introduced in fuzzy theory. As another resolution of type 2 fuzziness, we consider the problem involving a "fuzzy relation", which is a generalization of an ordinary binary relation.

In order to consider realistic modeling with fuzzy set concepts, the first part of this dissertation introduces fuzzy factors in scheduling problems, such as fuzzy due-dates, fuzzy processing times and fuzzy precedence constraints. The idea of fuzzy due-dates is based on decision maker's "degree of satisfaction" for the completion time of a job. In practice, the fuzzy due-dates are useful if due-dates are not rigid and slight violations for their "dead-lines" are accepted. The idea of fuzzy processing times is based on fuzzy number and is meaningful for the case in which processing times are treated as non-rigid values. For instance, such situations arise when we interpret persons (e.g., craftsmen) as machines in scheduling problems or when machines are unstable. The idea of fuzzy precedence constraints is based on fuzzy relation in fuzzy set theory. That is, in case a decision maker takes into account the precedence constraints corresponding to many factors (e.g., costs, technical constraints and others), it may not be enough to express them by only the ordinary (binary) relations.

As a concrete scheduling problem with these fuzzy factors, we then apply fuzzy due-dates, fuzzy processing times and fuzzy precedence constraints to the one machine scheduling problem. The multi-machine problems with fuzzy due-dates are also cosidered. These fuzzy scheduling problems are generalizations of the ordinary scheduling problems and realize more flexible formulations and realistic solutions based on a decision maker's subjective preference. Especially, as there exist conflict and complex factors in scheduling problems,

our approach is very useful.

In the second part of this dissertation, we investigate fuzzy network problems. That is, the idea of "degree of satisfaction" is applied to the sharing problem and the transportation problem, which are usually described on networks (graphs) with sources and sinks corresponding to supplies and demands, respectively. In the ordinary sharing problem, it is difficult to determine the values of "weights" (that represent relative importances of sinks) as unique values. Also, in the transportation problem, the amounts of some commodities to be transported from sources (interpreted as warehouses) to sinks (plants) may not be determined rigidly but may be stated in a fuzzy manner by the "parties concerned" (persons concerned with warehouses or plants). Accordingly, we consider them as decision making problems in the fuzzy environments with fuzzy weights, fuzzy supplies and fuzzy demands. As a successful "fuzzification" of them, it is pointed out, in particular, that the fuzzy transportation problem can be formulated and solved even though the total amount of demand values is larger than that of supply values.

The fuzzy combinatorial optimization problems proposed in this dissertation generalize the ordinary problems to achieve realistic modeling. We hope that the fuzzy approaches for combinatorial optimization will be extended to wider applications in many other areas.

Acknowledgement

The author would like to express his sincere appreciation to Professor Toshihide Ibaraki of Kyoto University for supervising this dissertation. His continuous encouragement and invaluable comments have helped to accomplish this dissertation.

The author is also heartily grateful to Professor Hiroaki Ishii of Osaka University for his enthusiastic discussions and persistent encouragement. He guided the author to the present study and have been giving the continuous and invaluable suggestions. Without his support, none of this work would have been possible.

The author highly indebted to Professor Emeritus Toshio Nishida of Osaka University, Professor at Osaka International University, for his invaluable comments and encouragement.

The author also wishes to thank Associate Professor Teruo Masuda of Okayama University for his helpful and stimulative discussions on fuzzy combinatorial optimization.

Furthermore, the author would like to thank Associate Professor Shogo Shiode and Dr. Shuichi Shinmori of Osaka University, Associate Professor Hideki Nishimoto of Kansai University and Dr. Hiroshi Morita of Kobe University for their helpful comments to complete this dissertation.

The author also wishes to his sincere thanks to all the members of Ryukoku University for their heartfelt support in working this research.

Finally, the author would like to express his sincere thanks to his parents Shingo Tada, Kimiko Tada, Takeo Inoue and Kiyoe Inoue for their continuous encouragement. The author smiles very special thanks to his wife Tomoe Tada and his son Rei Tada.

Contents

Chapter 1

INTRODUCTION

1.1 Purpose of the Dissertation and Historical Background

In order to deal with uncertainties of the phenomena in the real world, the theory of fuzzy sets originally poposed by Zadeh [Zad] is frequently used. Fuzzy Set Theory plays a vital role when we formalize and analyze the models in which imprecise and ambiguous factors are involved, in particular, for those which are accompanied by human cognitive process. This theory has been applied to a wide variety of fields, e.g.; control engineering, artificial intelligence, expert system, management science, pattern recognition, social science and so forth [Zim3].

In the field of Operations Research, the applications of fuzzy theory have been investigated particularly in relation to linear programming (i.e., "fuzzy mathematical programming") [Sak2, Inu]. In 1970, Bellman and Zadeh suggested a decision-making in a fuzzy environment by introducing a fuzzy goal (fuzzy objective function) and a fuzzy constraint, in which the decision was defined as the intersection of fuzzy goal(s) and fuzzy constraints [Bel]. This ap-

proach was applied for the first time to mathematical programming by Tanaka et al. [Tan1], and was also applied to linear programming by Zimmermann [Zim1]. After these, there are so many publications of fuzzy mathematical programming ([Zim2, Han, Orl, Tan2, Sak1 etc.]). In addition to mathematical programming, fuzzy set theory has been applied to the area of statistics, e.g., the so-called statistical data analysis such as linear regression analysis (for details; refer to [Isb]), statistical inference (estimation) (refer to [Oku]), clustering (refer to [Wat]) and so on.

As far as we know, however, there have been few applications of fuzzy set theory to combinatorial optimization problems. It appears therefore important to generalize the formulations of combinatorial optimization problems by introducing fuzzy set theory. The "fuzzification" of combinatorial optimization is needed for reasons of :

1. The standard solution methods may not fit to actual situations because of uncertain elements and non-rigid factors involved therein, e.g., human cognitive process.

2. Most of the combinatorial optimization probelms do not have efficient algorithms, and it may be possible to ease the situation by relaxing the restrictions in the sense of fuzzy set theory.

These two descriptions show different sides, i.e., which are based upon a decision maker's (abbreviated to "DM") two purposes: (1) "realistic" modeling and (2) solving "more efficiently". Although, of course, it is best to achieve them simultaneously, first we shall restrict our attention to the former purpose in this dissertation.

To achieve realistic modeling, we start with applications of fuzzy theory to some combinatorial problems such as scheduling and network flow problems,

for which comparatively efficient algorithms are known, but there is room for more "realistic" formulations. The "fuzzification" introduced in this dissertation is roughly classified into two types, i.e., *type 1: degree of satisfaction*- type and *type 2: uncertain data*-type.

The type 1 comes from the idea of fuzzy objective function proposed by [Bel], which is applied to combinatorial optimization problems. In other words, we generalize the problems by taking into account the difference of DM's subjectivity for a derived solution. For instance, concerning a certain maximization (minimization) problem, there may be a situation such that some DM is satisfied with a present solution even if its objective value is less (greater) than the "exact" optimal one, while another DM may stick to the exact optimum value. The former DM just wants to realize an objective value that is contained in his/her acceptance region of "subjective optimal values". Hence he/she treats the problem as that of maximizing (minimizing) the degree of satisfaction characterized by the idea of fuzzy objective function. This idea is introduced, in this dissertation, into the decision making in scheduling problems with fuzzy due-dates and network problems such as the transportation problem, the sharing problem and so forth. As a successful "fuzzification" of type 1, it is pointed out, in particular, that the fuzzy transportation problem can be formulated and solved even though the total amount of demand values is larger than that of supply values.

On the other hand, in order to deal with the characteristics of type 2, fuzzy sets are used to represent non-rigid (uncertain or ambiguous) data in combinatorial optimization problems. As a typical example of fuzzy data, we consider a processing time in scheduling models, since processing of a machine may not be rigid in some situation. In this case, we express the data by introducing an idea, called as a "fuzzy number", in fuzzy theory. As another

example of type 2, we consider the problem involving a "fuzzy relation", which is a generalization of an ordinary binary relation. The idea of fuzzy precedence constraints in this dissertation is based on fuzzy relation.

The main purpose of this dissertation is to lay a foundation of "fuzzy combinatorial optimization". That is, we try to generalize the classical combinatorial optimization problems such as scheduling problems and network flow problems by introducing fuzzy set theory.

In the rest of this chapter, we briefly describe basic concepts of fuzzy set theory, which will become necessary in the subsequent discussion, and review scheduling problems and network problems so that the generalizations of these problems in terms of fuzzy set theory can be precisely defined.

1.2 Related Areas of Fuzzy Theory

First, this section provides basic definitions of fuzzy sets and arithmetic operations with fuzzy numbers, which are to represent uncertain numbers, and after that, additional concepts called as an agreement index and a fuzzy relation are discussed.

1.2.1 Fuzzy sets

A fuzzy set is a generalization of the mathematical concept of a set. The word "crisp" is used to mean "non-fuzzy" in case that we need to distinguish ordinary sets from fuzzy sets.

Let E be a referential set (e.g., set of real numbers or set of integers). In classical set theory, given a crisp subset A of E, each element $x \in E$ satisfies either x belongs to A, or x does not belong to A. The subset A is represented

by its characteristic function $f_A(x) \in \{0,1\}$, namely

$$f_A(x) = \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

The crisp set A is generalized by introducing a function $\mu_{\boldsymbol{A}}$, called the *membership function*, which takes its values in the interval [0,1] instead of in the binary set {0,1} , i.e.,

$$\forall x \in E: \qquad \mu_{\boldsymbol{A}}(x) \in [0,1], \tag{1.1}$$

where the boldface letter $\boldsymbol{A}$ denotes a *fuzzy set* (strictly speaking, a *fuzzy subset*). That is, an element x of E belongs to $\boldsymbol{A}$ with a level $\mu_{\boldsymbol{A}}(x)$, which is located in $[0,1]$. The expression makes sense when we have a certain interpretation of this letter (see the following example).

Example 1.1

For instance, when we express a set of "young people" by a crisp subset, we can not define a "boundary age" x_b between "young" ages and "non-young" ages (in spite of no doubt that the ages seven, twelve and sixteen are young, whereas the ages thirty and forty are less young). Hence we may define it by introducing the following membership function $\mu_{\boldsymbol{A}}$:

$$\mu_{\boldsymbol{A}}(x) = \begin{cases} 1 & \text{if } 0 \le x \le 20, \\ (50-x)/30 & \text{if } 20 < x < 50, \\ 0 & \text{if } 50 \le x, \end{cases}$$

where we judge subjectively that $\mu_{\boldsymbol{A}}(20) = 1$ and $\mu_{\boldsymbol{A}}(50) = 0$, i.e., the ages twenty and fifty are young and not young, respectively. For intermediate ages, e.g., the age thirty is interpreted as a "fairly" young age (in view of the fact that $\mu_{\boldsymbol{A}}(30) = 0.6666\cdots$). ∎

Now we extend basic set-theoretic operations for fuzzy sets, i.e., the operations *intersection*, *union* and *complement* concerning fuzzy sets are defined as follows.

Definition 1.1 *The membership function $\mu_{\boldsymbol{I}}(x)$ of the intersection $\boldsymbol{I} = \boldsymbol{A} \cap \boldsymbol{B}$ is pointwisely defined by*

$$\mu_{\boldsymbol{I}}(x) = \min\left\{\mu_{\boldsymbol{A}}(x), \mu_{\boldsymbol{B}}(x)\right\}, \ x \in E.$$

Definition 1.2 *The membership function $\mu_{\boldsymbol{U}}(x)$ of the union $\boldsymbol{U} = \boldsymbol{A} \cup \boldsymbol{B}$ is pointwise defined by*

$$\mu_{\boldsymbol{U}}(x) = \max\left\{\mu_{\boldsymbol{A}}(x), \mu_{\boldsymbol{B}}(x)\right\}, \ x \in E.$$

Definition 1.3 *The membership function of the complement of a fuzzy set $\boldsymbol{A}$, $\mu_{\boldsymbol{C}}(x)$ is defined by*

$$\mu_{\boldsymbol{C}}(x) = 1 - \mu_{\boldsymbol{A}}(x), \ x \in E.$$

1.2.2 Fuzzy numbers

We introduce a fuzzy number and discuss some properties of fuzzy sets. Let α be a real parameter in the interval [0,1]. An *α-cut* (or *α-level set*) of a fuzzy set $\boldsymbol{A}$, denoted by A_α, is defined to be a crisp set:

$$A_\alpha = \left\{x \mid \mu_{\boldsymbol{A}}(x) \geq \alpha\right\}, \quad \alpha \in [0,1]. \tag{1.2}$$

We also call this an *interval of confidence* for the level of presumption α.

Conversely, a fuzzy set $\boldsymbol{A}$ is interpreted as a family of α-cuts (i.e., $\{A_\alpha\}_{\alpha\in[0,1]}$). That is, given a family of crisp sets A_α that satisfies the condition of monotonicity: $A_{\alpha_1} \supseteq A_{\alpha_2}$ if $\alpha_1 \leq \alpha_2$ (where $\alpha_1, \alpha_2 \in [0,1]$), we can define a fuzzy set $\boldsymbol{A}'$ by

$$\mu_{\boldsymbol{A}'}(x) = \sup\{\alpha \mid x \in A_\alpha, \ 0 \leq \alpha \leq 1\}. \tag{1.3}$$

If the maximum of the above α exists for each $x \in R$ (R : set of real numbers), we have $\mu_{\boldsymbol{A}}(x) = \mu_{\boldsymbol{A}'}(x)$ and hence $\boldsymbol{A} = \boldsymbol{A}'$, showing that $\boldsymbol{A}$ can be restored from α-cuts A_α.

To indicate that a fuzzy set can be regarded as a family of its α-cuts, the following expression (called *resolution identity*) is sometimes used.

$$\boldsymbol{A} = \bigcup_{\alpha} \alpha \cdot A_\alpha. \tag{1.4}$$

Now we describe two definitions in order to introduce a fuzzy number, which is useful to represent an uncertain number. First, the *convexity* of a fuzzy set are defined by using A_α as follows.

Definition 1.4 *A fuzzy set $\boldsymbol{A} \subset E$ is convex if and only if A_α is convex for any $\alpha \in [0,1]$, i.e., for arbitrary two points a and b in A_α, their convex sum*

$$c_\lambda = \lambda a + (1-\lambda) b,$$

also belongs to A_α for any λ such that $0 \leq \lambda \leq 1$.

Alternatively (as another definition), a fuzzy set $\boldsymbol{A}$ is *convex* if the following inequality holds for any $x, y \in R$ and any parameter $\lambda \in [0,1]$,

$$\mu_{\boldsymbol{A}}(\lambda x + (1-\lambda) y) \geq \min\left\{\mu_{\boldsymbol{A}}(x), \ \mu_{\boldsymbol{A}}(y)\right\}. \tag{1.5}$$

Next we define the *normality* of a fuzzy set as follows.

Definition 1.5 *A fuzzy set $\boldsymbol{A} \subset E$ is normal if and only if the following holds:*

$$\max_{x \in E}\left\{\mu_{\boldsymbol{A}}(x)\right\} = 1. \tag{1.6}$$

Now we are ready to define a *fuzzy number*.

Definition 1.6 *A fuzzy number $\boldsymbol{A}$ is a fuzzy set of R such that it is convex and normal.*

Nowadays, however, the definition of fuzzy number is very often modified as follows [Zim3].

Definition 1.7 *A fuzzy number* $\boldsymbol{A}$ *is a convex normalized fuzzy set* $\boldsymbol{A}$ *such that*

1. *there exists exactly one* $a_m \in R$ *with* $\mu_{\boldsymbol{A}}(a_m) = 1$ (a_m *is called as the mean value of* $\boldsymbol{A}$),
2. *the membership function* $\mu_{\boldsymbol{A}}(x)$ *is piecewise continuous: an increasing function for* $x \leq a_m$ *and a decreasing function for* $x \geq a_m$.

In this dissertation, we adopt Definition 1.7 to represent a fuzzy number. Intuitively, we interpret a fuzzy number $\boldsymbol{A}$ as an ambiguous number which is approximately equal to a_m .

It is easy to show that a fuzzy number $\boldsymbol{A}$ can always be restored from its α-cuts as in (1.3). Now several operations concerning fuzzy numbers are introduced. First, the addition of fuzzy numbers are defined as follows. Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two fuzzy numbers, and A_α and B_α α-cuts. Then the sum of these intervals $A_\alpha = [a_1^{(\alpha)}, a_2^{(\alpha)}]$ and $B_\alpha = [b_1^{(\alpha)}, b_2^{(\alpha)}]$ is given by

$$A_\alpha + B_\alpha = [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}]. \tag{1.7}$$

By using the α-cuts, we can express the addition of fuzzy numbers by

(Addition)

$$\boldsymbol{A} + \boldsymbol{B} = \bigcup_\alpha \alpha \cdot [a_1^{(\alpha)} + b_1^{(\alpha)}, a_2^{(\alpha)} + b_2^{(\alpha)}]. \tag{1.8}$$

Similarly, other arithmetic operations are defined as follows:

(Subtraction)

$$\boldsymbol{A} - \boldsymbol{B} = \bigcup_\alpha \alpha \cdot [a_1^{(\alpha)} - b_2^{(\alpha)}, a_2^{(\alpha)} - b_1^{(\alpha)}], \tag{1.9}$$

(Multiplication)

$$\boldsymbol{A} \cdot \boldsymbol{B} = \bigcup_\alpha \alpha \cdot [a_1^{(\alpha)} \cdot b_1^{(\alpha)}, a_2^{(\alpha)} \cdot b_2^{(\alpha)}], \tag{1.10}$$

(Division)

$$\boldsymbol{A} \div \boldsymbol{B} = \bigcup_\alpha \alpha \cdot [a_1^{(\alpha)} \div b_2^{(\alpha)}, a_2^{(\alpha)} \div b_1^{(\alpha)}], \tag{1.11}$$

where (Addition, Subtraction) and (Multiplication, Division) are defined in the set of real numbers R and in the set of positive real numbers R^+, respectively.

Besides the above arithmetics for fuzzy numbers, we sometimes utilize the concept of *fuzzy upper bound* [Kau], called an *agreement index*, when two kinds of fuzzy sets such as the following $\boldsymbol{A}$ and $\boldsymbol{H}$ are used together. Let $\boldsymbol{A} \subset R$ be a fuzzy number such that

$$\mu_{\boldsymbol{A}}(x) = \begin{cases} I_A(x) & \text{if } a_1 \leq x \leq a_2, \\ 1 & \text{if } x = a_2, \\ D_A(x) & \text{if } a_2 \leq x \leq a_3, \end{cases} \tag{1.12}$$

where $I_A(x)$ and $D_A(x)$ denote arbitrary monotonically incereasing function and decreasing function, respectively, which satisfy that $I_A(a_1) = 0$, $I_A(a_2) = 1$, $D_A(a_2) = 1$ and $D_A(a_3) = 0$. Let $\boldsymbol{H} \subset R$ be a fuzzy set represented by its membership function $\mu_{\boldsymbol{H}}(x)$ such that

$$\mu_{\boldsymbol{H}}(x) = \begin{cases} 1 & \text{if } x \leq h_1, \\ D_H(x) & \text{if } h_1 \leq x \leq h_2, \\ 0 & \text{if } x \geq h_2, \end{cases} \tag{1.13}$$

where $D_H(x)$, which satisfies $D_H(h_1) = 1$ and $D_H(h_2) = 0$, denotes an arbitrary monotonically decreasing function.

This type of fuzzy set $\boldsymbol{H}$ is often used in decision making and the membership function $\mu_{\boldsymbol{H}}$ is interpreted as a function that represents the *degree of satisfaction* for some decision-maker (abbreviated to "DM"); if $\mu_{\boldsymbol{H}}(x) = 1$,

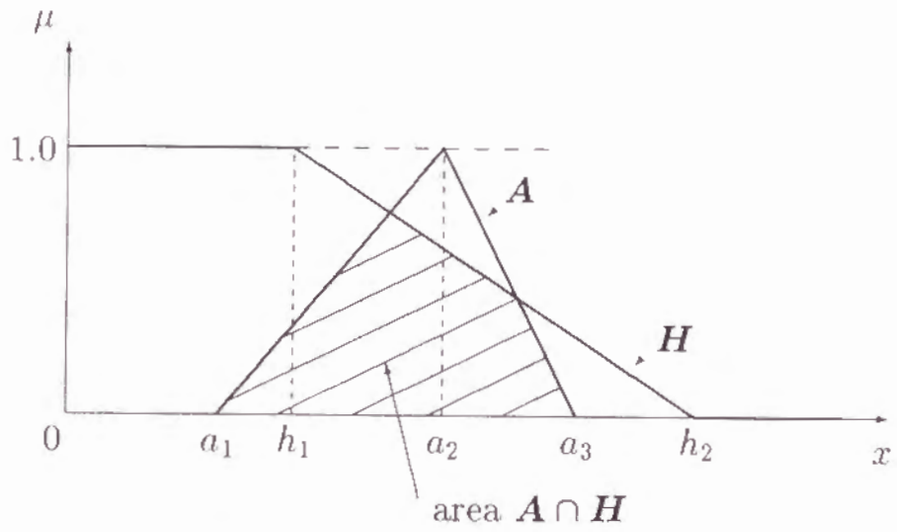

Fig. 1.1 Example of agreement index

DM is satisfied with the value x, while if $\mu_{\boldsymbol{H}}(x) = 0$, he is dissatisfied with the decision. For intermediate values, e.g., $\mu_{\boldsymbol{H}}(x) = 0.8$, we may say that he feels "quite" good. In this dissertation, this interpretation is applied to fuzzy sharing problem, fuzzy due-dates in scheduling problems and so on.

Now we define the *agreement index* as the area ratio of the intersection $\boldsymbol{A} \cap \boldsymbol{H}$ to the fuzzy number $\boldsymbol{A}$ (see Fig.1.1), where the area of a fuzzy set $\boldsymbol{B}$ is measured by

$$area\ \boldsymbol{B} = \int_R \mu_{\boldsymbol{B}}(x)dx.$$

Definition 1.8 *The agreement index of* $\boldsymbol{A}$ *with regard to* $\boldsymbol{H}$ *is a real number* $\iota \in [0,1]$ *given by*

$$\iota(\boldsymbol{A}, \boldsymbol{H}) = (area\ \boldsymbol{A} \cap \boldsymbol{H}) \Big/ (area\ \boldsymbol{A}). \tag{1.14}$$

When the membership function $\mu_{\boldsymbol{H}}(x)$ is used as the fuzzy objective function, there may be some situation in which the value x is not rigid but is given as a fuzzy number $\boldsymbol{A}$. In such situation, we may use the concept of agreement index ι in order to indicate his degree of satisfaction for the uncertain value $\boldsymbol{A}$.

1.2.3 Fuzzy relations

Generalizing an ordinary binary relation for a pair of two elements (i.e., either the pair has the relation or does not have it), its fuzzy version called a *fuzzy relation* can be defined. Let X and Y be crisp sets. For a pair of elements $x \in X$ and $y \in Y$, a fuzzy relation $\boldsymbol{R}$ on $X \times Y$ (i.e., the Cartesian product) is defined by

$$\boldsymbol{R} = \left\{ ((x,y), \mu_{\boldsymbol{R}}(x,y)) \mid (x,y) \subseteq X \times Y \right\}, \tag{1.15}$$

where $((x,y), \mu_{\boldsymbol{R}}(x,y))$ means the pair (x,y) is in a fuzzy relation $\boldsymbol{R}$ (i.e., $x\boldsymbol{R}y$) with its membership $\mu_{\boldsymbol{R}}(x,y)$. In other words, $\mu_{\boldsymbol{R}}(x,y)$ denotes the strength of relation $\boldsymbol{R}$ between x and y: a fuzzy relation $\boldsymbol{R}$ is a fuzzy set over $X \times Y$, i.e., $\mu_{\boldsymbol{R}} : X \times Y \to [0,1]$.

Example 1.2

Let X and Y be subsets of Z^+ (set of positive integers). We define, for $x \in X$ and $y \in Y$, a fuzzy relation $\boldsymbol{R}$: ="x is considerably larger than y". Its membership function may be given by

$$\mu_{\boldsymbol{R}}(x,y) = \begin{cases} 0 & \text{if } x \le y, \\ (x-y)/9y & \text{if } y < x \le 10y, \\ 1 & \text{if } 10y < x, \end{cases}$$

If X and Y are finite sets, we often use a matrix, called a *fuzzy relation matrix*, to represent a fuzzy relation. For example, for $X = \{x_1, x_2, x_3\} =$

$\{80, 7, 50\}$ and $Y = \{y_1, y_2, y_3, y_4\} = \{10, 1, 30, 7\}$, the fuzzy matrix may be given by :

$$\boldsymbol{R}: \quad \begin{array}{c} \\ x_1 \\ x_2 \\ x_3 \end{array} \begin{array}{c} \begin{array}{cccc} y_1 & y_2 & y_3 & y_4 \end{array} \\ \begin{bmatrix} 0.8 & 1 & 0.2 & 1 \\ 0 & 0.7 & 0 & 0 \\ 0.4 & 1 & 0.1 & 0.7 \end{bmatrix} \end{array},$$

where the i-th row and j-th column correspond to $x_i \in X$ and $y_j \in Y$, respectively. The (i, j)-th element indicates $\mu_{\boldsymbol{R}}(x_i, y_j)$, e.g., $\mu_{\boldsymbol{R}}(x_3, y_4) = 0.7$.

■

Finally, we may interpret a fuzzy relation as a graph which is defined on $\mathcal{N} \times \mathcal{N}$, where $\mathcal{N}$ is a set of nodes, and the fuzzy relation corresponds to the existence of arcs in the traditional graph theory (for details of graph or network theory; refer to Section 1.4).

Definition 1.9 *A fuzzy (directed) graph $\boldsymbol{G}$ is defined by*

$$G(x, y) = \left\{ ((x, y), \mu_{\boldsymbol{R}'}(x, y)) \mid (x, y) \in \mathcal{N} \times \mathcal{N} \right\},$$

where N is a finite set of nodes.

For a fuzzy relation $\boldsymbol{R}$ defined on the Cartesian product of two distinct sets X and Y, symmetric law is usually assumed, i.e., $\mu_{\boldsymbol{R}}(x, y) = \mu_{\boldsymbol{R}}(y, x)$ for any $x \in X$ and $y \in Y$. However the above fuzzy relation $\boldsymbol{R}'$ defined on $\mathcal{N} \times \mathcal{N}$ may not be symmetric, thereby necessitating the introduction of a "directed" graph. We shall focus our attention on directed fuzzy graphs in this dissertation. This generalization will be subsequently applied to scheduling problems with fuzzy precedence constraints (relations).

1.3 Review of Scheduling Problems

1.3.1 Scheduling problems

Scheduling theory has been applied to compute schedules in various systems arisen in industrial production, processing on computers, traffic facilities and so forth. There are deterministic and stochastic approaches to scheduling problems. Although stochastic approaches are important in order to deal with some systems that include stochastic factors [Bła,Moh], we mainly consider *deterministic* scheduling problems (that is, it is assumed that all the parameters are deterministically known a priori) in this dissertation, and introduce the concept of fuzzy sets to represent non-crisp aspects of the problems.

In general, a *scheduling problem* is stated as follows. Given a set of n *jobs* $J_1, J_2, \cdots, J_n$ to be processed on a set of available *machines*, it is asked to obtain an optimal schedule that minimizes a certain objective function under given processing conditions. Here, each of the jobs may consist of more than one *operation*. To complete a job, we must process all of its operations. However, for a brief description of a schedule, we shall suppose for a while that each job consists of exactly one operation. In this case, we do not distinguish "job" and "operation". The word "schedule" means an assignment of jobs to machines and the determination of a processing order of jobs on each machine. Hence seeking a schedule results in the following:

1. Is each job processed on which machine and in what order?

2. When does each machine start processing the assigned jobs?

In actual applications, jobs and machines may be interpreted as computer tasks (programs) and processors (computers), ships and dockyards, classes and teachers, patients and hospital equipments and so on.

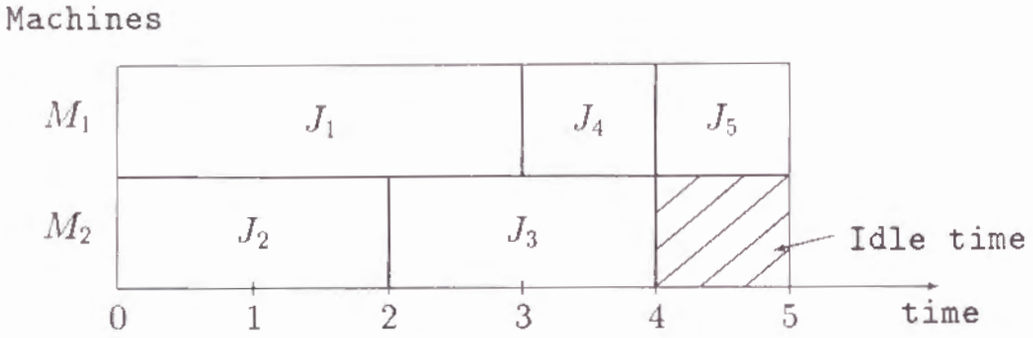

Fig. 1.2 Example of Gantt chart

To comprehend schedules intuitively, let us introduce a *Gantt chart*, which graphically represents the machine assignment and processing time of all jobs J_i (here it is assumed that each job consists of one operation). For example, the Gantt chart in Fig.1.2 indicates that jobs $J_i\,(j = 1, \cdots, 5)$ are processed on machines M_1 and M_2 in the order of $J_1 \rightarrow J_4 \rightarrow J_5$ and $J_2 \rightarrow J_3$, respectively. We can easily see that, for example, the completion times of jobs J_1, J_3 are at 3, 4, respectively, and the completion time of all five jobs is at 5.

1.3.2 Classification and optimal criteria

In classifying scheduling problems, we have to first specify the characteristics of machines, whereby scheduling problems are classified, as shown in Fig.1.3. First, they are distinguished between *one machine* and "multi" machines (i.e., more than one machine) which are either *parallel*, i.e., all machines have the same function, or *dedicated*, i.e., machines are specialized to the execution of certain jobs. In the case of parallel machines, each job, which consits of one operation, may be processed by any machine, while in the case

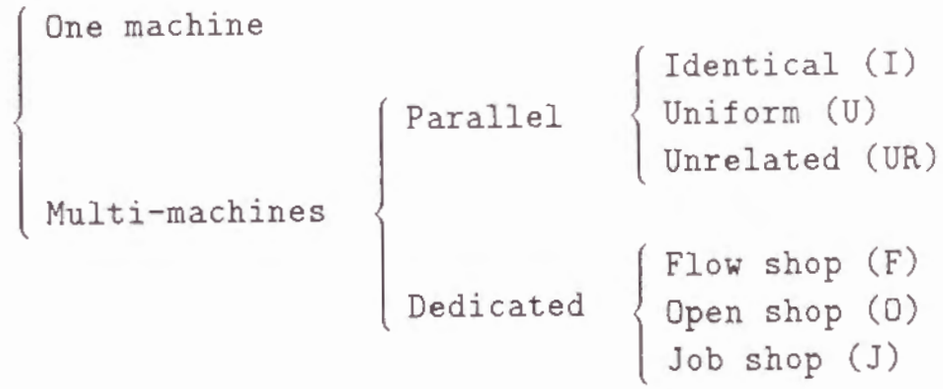

Fig. 1.3 Classification of scheduling

of dedicated-type, where we assume that each job generally consists of more than one operation, the machine used to process each operation is specified beforehand. We distinguish "job" and "operation" from now onward.

Further, parallel machines are classified into three types according to their relative speeds, that is, *identical* machines if all machines have equal job-processing speeds, *uniform* if they have different speeds but the speed of each machine is a given constant that does not depend on jobs, and *unrelated* if the machine speeds depend on the particular jobs to be processed.

The dedicated machines are further classified according to the ordrer of machines used to process the operations of each job, where we assume that each job consists of a set of operations and the machine that processes each operation is specified a priori. It is called a *flow shop* if the order of machines that process all operations of each job is the same for all jobs, a *job shop* if the specified order of machines may be different for each job, and an *open shop* if the order of machines is not specified for any job.

To formulate a scheduling problem formally, the following data are usually

given for each job $J_i \in \mathcal{J}$, where $\mathcal{J}$ denotes a set of n jobs $\{J_1, J_2, \cdots, J_n\}$.

1. *Processing time* : $p_i = [p_{i1}, p_{i2}, \cdots, p_{im}]$, where p_{ij} is the time needed by machine $M_j \in \mathcal{M}$ to complete J_i and $\mathcal{M}$ is a set of m machines $\{M_1, M_2, \cdots, M_m\}$. In the case of one machine or identical machines, we have $p_{ij} = p_i$ for all $j = 1, \cdots, m$. If the machines are uniform, then $p_{ij} = p_i/s_j$, where p_i denotes the *standard processing time* (usually on the slowest machine) of J_i and, s_j (usually $s_1 = 1$ and $s_j \geq 1$ for $j \geq 2$) is the speed factor of machine M_j. In the case of dedicated machines, p_{ij} is the time needed by M_j to complete O_{ij}, which is the operation of the $i-$th job to be processed by the $j-$th machine.

2. *Ready time* : A ready time (or an *arrival time*) r_i is the time at which J_i is ready for processing. In this dissertation, it is always assumed that $r_i = 0$ for all $i = 1, \cdots, n$, that is, all jobs are *available* at time 0.

3. *Due-date* : a due-date d_i (i.e., so-called "deadline") denotes that it is demanded that the processing of job J_i is completed by time d_i.

Now we describe a formal definition of schedule in detail. A *schedule* is an assignment of macines from set $\mathcal{M}$ to jobs from set $\mathcal{J}$ along the time horizon under the following restrictions.

1. At most one machine can process each job at the same time.

2. Each machine can process at most one job at the same time.

The condition 1 is always satisfied in the case of one machine problem. In the case that each job consists of more than one operations, however, we must take account of some additional conditions according to machine-types.

When jobs are to be processed on dedicated machines, each job J_i is divided into k operations $O_{i1}, O_{i2}, \cdots, O_{ik}$, where k shows the number of operations per job. In a flow shop system, k is equal to m (the number of machines) and their order of processing is such that O_{i1} is processed on M_1, O_{i2} on $M_2, \cdots$, and O_{ik} on $M_k (= M_m)$. Moreover the processing of $O_{i,j-1}$ must always precede the processing of $O_{ij}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, m$. In the case of an open shop system, although each job is also devided into m operations with the same allocation of the operatrions to machines, the order of processing operations is not specified a priori. In a job shop system, the number of operations per joh, their assignment to machines and the order of their processing are arbitrary (but specified a priori).

Moreover, we sometimes assume that "preemption" are allowed. That is, the mode of processing is called *preemptive* if processing a job (an operation in the case of dedicated machines) may be interrupted at any time and restarted later (perhaps on another machine) at no cost. If the preemption of any job (operation) is not allowed, we call it *nonpreemptive.*

Furthermore, there may be a constraint on the order of processing jobs, such as processing of J_j can be started only after completing the processing of J_i. This is denoted by $J_i < J_j$ and is called a *precedence constraint.* The jobs in $\mathcal{J}$ are called *dependent* if there is a nonempty set of precedence constraints, otherwise the jobs are *independent.*

In formulating and solving a scheduling problem, it is important how we define the optimality of schedules. To evaluate schedules, we mainly use the following two criteria $C_{\max}$ and $L_{\max}$ in this dissertation, though other criteria are also used in applications.

- *Maximum completion time* : $C_{\max} = \max\{C_i\}$,
  where C_i denotes the completion time of J_i, $i = 1, \cdots, n$.

- *Maximum lateness* : $L_{\max} = \max\{L_i\}$, where $L_i = C_i - d_i, i = 1, \cdots, n$.

- *Maximum tardiness* : $T_{\max} = \max\{T_i\}$, where $T_i = \max\{C_i - d_i, 0\}, i = 1, \cdots, n$.

- *Mean flow time* : $\bar{F} = \frac{1}{n}\sum_{i=1}^{n} F_i$, where $F_i = C_i - r_i$ denotes the *flow time* of job J_i.

- *Number of tardy jobs* : $N_T = \sum_{i=1}^{n} u(T_i)$, where $u(x) = 1$ if $x > 0$, and $u(x) = 0$ otherwise.

By combining of the above machine types and optimality criteria, it is possible to define many scheduling problems. To denote such scheduling problems compactly, a four-field notation $\alpha \mid \beta \mid \gamma \mid \delta$ is commonly used. The first letter α describes the number of jobs to be processed. We usually assume that $\alpha = n$, where n is an arbitrary positive integer. The second and third letters β and γ indicate the number of machines and the type of machines, respectively. If $\beta = 1$ (i.e., one machine), we omit the third letter γ (i.e., three-field notation $\alpha \mid \beta \mid \delta$). Otherwise, the corresponding symbol shown at the right end of Fig.1.3 (e.g., I: identical parallel, O: open shop etc.) is used as the third letter. The last letter δ is the optimality criterion such as $L_{\max}$: minimization of maximum lateness and $C_{\max}$: minimization of maximum completion time.

1.3.3 Specification of some scheduling problems

Among the various scheduling problems as introduced in the previous section, we define below three particular scheduling problems in detail.

A One machine scheduling problem : As a typical deterministic one machine scheduling problem, we define here the problem of minimizing the maximum lateness, i.e., $n \mid 1 \mid L_{\max}$, which is described as follows.

1. A given set of jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ is processed on one machine.
2. Processing time of each job J_i is $p_i, i = 1, \cdots, n$.
3. Each job $J_i, i = 1, \cdots, n$ has a deterministic due-date d_i.
4. Preemptions are not allowed.
5. The objective is to find a schedule that minimizes the maximum lateness $L_{\max}$.

It is known [Smi] that an optimal schedule, which minimizes the value of the optimality criterion $L_{\max}$, can be found by arranging jobs in nondecreasing order of their due-dates, i.e., by the well-known *EDD (Earliest Due Date)-rule*.

In Chapter 3 of this dissertation, we generalize the problem by introducing *fuzzy due-dates* in order to represent that due-dates are not rigid and some violations may be accepted in some situations. We also consider other fuzzy versions, in the same chapter, such as one machine problems with fuzzy processing times and with fuzzy precedence relations.

B Identical parallel machine scheduling problem : For the scheduling problem of parallel machines $n \mid m \mid I \mid L_{\max}$, we introduce a *general due-date* d_{ij}. That is, each job $J_i, i = 1, \cdots, n$ has a due-date d_{ij} for each machine $M_j, j = 1, \cdots, m$ (i.e., due-date is dependent upon machines).

This general due-date can be explained as follows. Assume that a factory "A" wants to recieve a completed joh J_i from machine M_j. However, the

delivery time of J_i to A may depend on the machine M_i from which J_i is sent out. In this case, the acutual due-date of J_i at machine M_j must be determined by taking into account this delivery time. To capture this effect, we generalize so that a job may have different due-dates corresponding to machines.

The problem $n \mid m \mid I \mid L_{\max}$ with general due-dates is described as follows.

1. A given set of jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ is processed on a set of identical parallel machines $\mathcal{M} = \{M_1, M_2, \cdots, M_m\}$.

2. Processing time of each job J_i is $p_i, i = 1, \cdots, n$.

3. Each job $J_i, i = 1, \cdots, n$ has a deterministic due-date d_{ij} for each machine $M_j, j = 1, \cdots, m$.

4. Preemptions are allowed.

5. Tbe objective is to find a schedule that minimizes the maximum lateness $L_{\max}$.

For this scheduling problem $n \mid m \mid I \mid L_{\max}$ with general due-dates, we investigate its fuzzy version, i.e., *fuzzy general due-dates*. It will be shown in Section 4.1 that the problem can be solved by introducing a certain network and the idea of modified due-dates, and combining them with a binary search technique.

C Two machine open shop scheduling problem : Among various shop-type scheduling problems, we consider the two machine open shop scheduling problem that minimizes the maximum completion time, i.e., $n \mid 2 \mid O \mid C_{\max}$. This problem $n \mid 2 \mid O \mid C_{\max}$ can be solved by the algorithm of Gonzalez and Sahni [Gon]. The problem is then generalized by allowing that

the speed of each machine is controllable. That is, we consider that a job J_i $(i = 1, \cdots, n)$ on M_j $(j = 1, 2)$ can be processed faster by increasing its speed from its "standard" speed, by paying the cost that is proportional to its speed [Ish3,Tad1]. Thus, the objective is to determine an optimal schedule and jobwise speed of the machines simultaneously, such that the total sum of the cost associated with a certain optimality criterion and the cost required to attain jobwise machines speed is minimized.

The generalized problem is defined as follows.

1. There are two machines M_1, M_2 and n jobs $J_i, i = 1, 2, \cdots, n$, each of which consists of two operations O_{i1} and O_{i2}. All jobs are open shop type, i.e., the processing order of its two operations is not specified.

2. Speed of each machine is controllable.

3. "Processing amounts" (standard processing times at unit speed) of two operations O_{i1} and O_{i2} of each job $J_i, i = 1, \cdots, n$ are a_i and b_i, respectively.

4. Each job $J_i, i = 1, \cdots, n$ has a deterministic due-date d_i.

5. Each machine processes at most one job at a time, and each job is processed on at most one machine at a time.

6. Preemptions are allowed.

The problem is then further generalized in Chapter 4 by introducing fuzzy due-dates.

1.4 Review of Network Problems

1.4.1 Network problems

The solution techniques for network flow problems are utilized in many areas of combinatorial optimization problems. In general, the *network flow problem* is described as determining a maximal steady state flow from a source node to a sink node subject to capacity constraints on arcs [For]. Although two words "network" and "graph" are often used as a synonym, we distinguish them in this dissertation by using "network" if there exist flows in arcs and otherwise "graph".

The formal definition of a network is as follows. A *network* $G = [\mathcal{N}; \mathcal{A}]$ consists of two sets, a set of nodes $\mathcal{N}$ and a set of arcs $\mathcal{A}$, where each arc is an ordered pair of nodes taken from $\mathcal{N}$. An arc (x, y) is usually represented by an arrow directing from node x to node y. In this dissertation, some fuzzy versions of the network problems will be investigated. A particular kind of network $BG = [S, T; \mathcal{A}]$ called a *bipartite* network (or graph) is also considered, in which node set $\mathcal{N}$ is partitioned into two disjoint subsets S and T, and the arcs of $\mathcal{A}$ are directed from the nodes in S (interpreted as *supply*) to the nodes in T (interpreted as *demand*).

Network problems were historically first applied to transportation and communication networks. In recent years, the application areas have been extended to agriculture, finance, marketing, warehousing, and so on. In addition, stochastic approaches are often taken when we consider some stochastic factors in networks [Deg,Ish1]. As an alternative approach to the realistic models, this dissertation proposes some applications of fuzzy theory to network problems.

1.4.2 Maximal flow problem

Among various network problems, the maximal flow problem is the most fundamental. Given a network $G = [\mathcal{N}; \mathcal{A}]$, suppose that each arc $(x, y) \in \mathcal{A}$ has the associated nonnegative *capacity* $c(x, y)$. We are interested in finding the maximal amount of flow which we can send from a *source* node to a *sink* node. We call such a flow a *maximal network flow* or simply a *max-flow*.

To formulate the max-flow problem, further notations are introduced. Let s and t denote the source and sink nodes, respectively. Other nodes are called *intermediate*. We use notation $f(x, y)$ to denote the flow value in arc (x, y).

Using these notations, the max-flow problem is formulated as follows:

$$MFP: \quad \text{Maximize} \quad v, \tag{1.16}$$

subject to

$$\sum_{y \in A(x)} f(x,y) - \sum_{y \in B(x)} f(y,x) = \begin{cases} v & \text{if } x = s, \\ 0 & \text{if } x \neq s, t, \\ -v & \text{if } x = t, \end{cases} \tag{1.17}$$

$$0 \leq f(x,y) \leq c(x,y), \qquad \forall (x,y) \in \mathcal{A}, \tag{1.18}$$

where $A(x)$ and $B(x)$ ("after x" and "before x") denote the sets of all nodes $y \in \mathcal{N}$ such that (x, y) are arcs out of x and such that (y, x) are arcs into x, respectively. The equations (1.17) denote that if x is equal to source s then the net flow out of the source is v, that if x is equal to sink t then the net flow out of t is $-v$ (i.e., the net flow into t is v), and that if x is an intermediate node, the net flow out of it is zero (therefore, equations (1.17) for intermediate nodes are called *flow conservation constraints*). For an intermediate node x, the total amount of flow out of x is equal to the total amount flow sent into x. For a flow f satisfing (1.17) and (1.18), its v is called the *flow value* of f. Therefore, the max-flow problem asks to obtain the maximum flow value. The recognition of a certain subset of arcs, called a *cut*, often plays an important

role in solving the problems of this kind. A *cut* $\mathcal{C}$ in $[\mathcal{N};\mathcal{A}]$ *separating* s *and* t is a set of arcs $(X,\overline{X}) = \{(x,y) \in \mathcal{A} \mid x \in X \text{ and } y \in \overline{X}\}$, where $s \in X$ and $t \in \overline{X}$, and $\overline{X} = \mathcal{N} - X$. The capacity of a cut $(X,\overline{X})$ is defined by

$$c(X,\overline{X}) = \sum_{(x,y)\in(X,\overline{X})} c(x,y). \tag{1.19}$$

Let $x_1,\cdots,x_n$ $(n \geq 2)$ be a sequence of distinct nodes such that there exists an arc $(x_i,x_{i+1}) \in \mathcal{A}$ for each node x_i $(i = 1,\cdots,n-1)$. Then the sequence of nodes and arcs $x_1,(x_1,x_2),x_2,\cdots,(x_{n-1},x_n),x_n$ is called a *chain* from x_1 to x_n. It is immediate to see that any chain from s to t must contain some arc of every cut $\mathcal{C} = (X,\overline{X})$. Therefore, if all arcs of a cut $\mathcal{C}$ were deleted from the network, there would be no chain from s to t and the maximal flow value for the resulting network would be zero. From this observation, we conclude that the value v of a flow f cannot exceed the capacity of any cut $\mathcal{C}$. The converse of this property is also true as stated in the following well-known theorem (the formal proof is omitted; see [For]).

Theorem 1.1 (Max-flow min-cut theorem) *For any network* $[\mathcal{N};\mathcal{A}]$*, the maximum flow value from* s *to* t *is equal to the minimum capacity of all cuts that separate* s *and* t*.*

1.4.3 Specification of some network problems

We describe here some (standard) network problems, fuzzy versions of which will be investigated in Chapters 5 through 7 of this dissertation.

A Sharing problem : The sharing problem, originated by Brown [Bro], is defined on a *capacitated* network $G = [\mathcal{N};\mathcal{A}]$, i.e., each arc $(x,y) \in \mathcal{A}$ has its capacity $c(x,y) > 0$. The node set $\mathcal{N}$ includes a set of source nodes $S = \{s_1,\cdots,s_m\}$ and a set of sink nodes $T = \{t_1,\cdots,t_n\}$, where m and

n are positive integers. Apart from these, there are a *super source* s and a *super sink* t, respectively, in this problem, where there are non-capacitated (i.e., capacity of infinity) arcs from s to all the source nodes $s_1,\cdots,s_m$ and there are non-capacitated arcs from all the sink nodes $t_1,\cdots,t_n$ to t. We use the extended network $G' = [N';A']$ ($N' = \mathcal{N} + \{s,t\}$ and $A' = \mathcal{A} + \{(s,s_1),(s,s_2),\cdots,(s,s_m)\} + \{(t_1,t),(t_2,t),\cdots,(t_n,t)\}$) and consider the max-flow problem from s to t.

The aim of the sharing problem is to find an equitable distribution of flows (i.e., resource) supplied from source nodes to sink nodes. In other words, it is a determination of flow values to be sent equitably into t_j, i.e., we must revise each value of $c(t_j,t)$'s (which are initially equal to ∞), before applying a method of the max-flow problem. To formulate this problem, a trade-off function $TO(f(t_j))$ is introduced, where $f(t_j)$ denotes the total amount of flow into a sink node t_j, i.e.,

$$f(t_j) = \sum_{x\in B(t_j)} f(x,t_j). \tag{1.20}$$

Each t_j has its relative weight $w(t_j)$ and it gives a *trade-off value* defined as the quotient of $f(t_j)$ and $w(t_j)$. The objective is to maximize the smallest value among all the trade-off values:

$$SP: \quad \text{Maximize} \quad \min_{t_j\in N'} \{f(t_j)/w(t_j)\} \tag{1.21}$$

subject to

$$\sum_{t_j\in T} f(t_j) = v^*, \tag{1.22}$$

$$\sum_{x\in N'-\{t\}} f(x,y) = \sum_{z\in N'-\{s\}} f(y,z),\ y \in \mathcal{N}, \tag{1.23}$$

$$0 \leq f(x,y) \leq c(x,y),\ x \in N' - \{t\},\ y \in N' - \{s\}, \tag{1.24}$$

where v^* denotes the total amount of flows supplied from s_i's.

B Transportaion problem : The transportation problem is a minimum cost flow problem on a bipartite network $BG = [S, T; \mathcal{A}]$, where S and T denote a set of m supply (or plant) nodes and a set of n demand (or warehouse) nodes, respectively (i.e., $\mid S \mid = m$ and $\mid T \mid = n$), and $\mathcal{A}$ is a set of arcs (x, y) such that $x \in S$ and $y \in T$. Each arc $(x, y) \in \mathcal{A}$ has a *flow cost* c_{xy} per unit flow in arc (x, y). The *transportation cost* from x to y is defined as the product of the flow $f(x, y)$ and its cost c_{xy}.

The goal of this problem is to determine a flow such that the total transportation cost is minimized among all the possible flows from S to T. The problem is formulated as follows:

$$TP: \quad \text{Minimize} \quad \sum_{x \in S, y \in T} c_{xy} f(x, y) \tag{1.25}$$

$$\text{subject to} \quad \sum_{y \in T} f(x, y) = a_x, \ x \in S, \tag{1.26}$$

$$\sum_{x \in S} f(x, y) = b_y, \ y \in T, \tag{1.27}$$

$$0 \leq f(x, y), \ x \in S, y \in T, \tag{1.28}$$

where a_x and b_y denote the total amounts of supply from x and demand to y, respectively.

An optimal flow of this problem is seeked by the modified primal simplex method for the "LP" problem (since this problem is a special case of linear programming problem). One of such algorithms is called the *stepping stone method*.

1.5 Outline of the Dissertation

This dissertation consists of two parts, PART I concerning fuzzy scheduling problems and PART II concerning fuzzy network probelms.

PART I comprises three chapters; Chapter 2 through Chapter 4. In Chapter 2, we introduce some fuzzy factors such as fuzzy due-dates, fuzzy processing times and fuzzy precedence constraints in scheduling models, and give their numerical examples in order to understand these fuzzy factors clearly. The idea of fuzzy due-dates is based on the concept of DM's (decision-maker's) "degree of satisfaction" for completion times of jobs, while fuzzy processing times are fuzzy numbers and fuzzy precedence constraints are fuzzy relations.

Chapter 3 discusses fuzzy scheduling problems on one machine. Since it is generally easier to treat the problems on one machine than on multi-machines (i.e., two, three and so on), we begin with one machine problem $n \mid 1 \mid L_{\max}$ with fuzzy due-dates as a first step of fuzzy scheduling. Besides this problem, the same chapter also treats one machine scheduling problems with fuzzy processing times and fuzzy precedence constraints.

In Chapter 4, two fuzzy scheduling problems on multi-machines are discussed. One is the problem on identical machines $n \mid m \mid I \mid L_{\max}$ with fuzzy due-dates. After extending the standard problem to a general due-dates version such that each job has its own due-date for each machine, we introduce its fuzzy version. The other is the two machine open shop problem $n \mid 2 \mid O \mid L_{\max}$ with fuzzy due-dates. This problem is first generalized by allowing each machine speed to be controllable, and then further generalized by introducing fuzzy due-dates.

PART II of this dissertation consists of Chapters 5 and 6 discusses the fuzzy network problems. That is, the idea of "degree of satisfaction" is applied to the sharing problems as fuzzy weights ("weights" in the ordinary problem represent relative importances of sinks) in Chapter 5, and the idea is applied to the transportation problem as fuzzy supplies and fuzzy demands (the values of supplies and demands are rigidly given in the ordinary problem) in Chapter

6.

Finally, Chapter 7 summarizes the contribution of this dissertation and discusses future research topics.

Part I

FUZZY SCHEDULING PROBLEMS

Chapter 2

FUZZY FACTORS IN SCHEDULING MODELS

2.1 Fuzzy Due-Dates

In the maximum lateness (or tardiness) scheduling problems, due- dates are usually given as rigid values. In some situation, however, due-dates are not rigid and slight violations for their "dead-lines" may be accepted. In order to consider a realistic scheduling problems in such a situation, we introduce *fuzzy due-dates* characterized by the corresponding membership functions, which show DM's (some decision maker's) "degree of satisfaction" for completion times of jobs. In this case, we can represent the differences in DM's treatment of tardiness by the shapes of the membership functions (see the following example).

We assume for simplicity that the membership function $\mu_{\boldsymbol{D}_i}(i = 1, \cdots, n)$ of completion time C_i associated with each job J_i is piecewise linear as given in the following :

$$\mu_{\boldsymbol{D}_i}(C_i) = \begin{cases} 1 & \text{if } C_i \leq d_i, \\ 1 - \dfrac{C_i - d_i}{e_i} & \text{if } d_i < C_i \leq d_i + e_i, \\ 0 & \text{if } C_i > d_i + e_i, \end{cases} \tag{2.1}$$

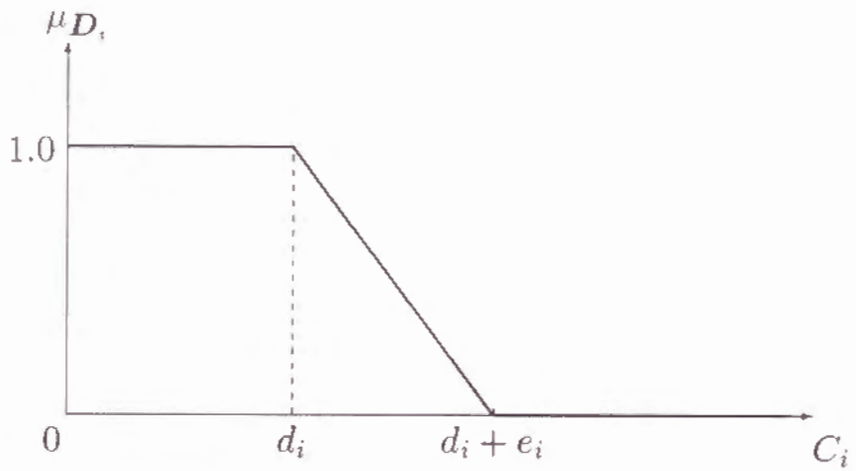

Fig. 2.1 Membership function $\mu_{\boldsymbol{D}_i}(C_i)$

where d_i and e_i are positive constants. The value of d_i corresponds to the original due-date of J_i. Consequently if the completion time C_i is not greater than d_i, DM is satisfied completely, i.e., the degree of satisfaction is 1. But if late, the degree decreases according to the value of C_i, and the degree is 0 when the completion time is not less than $d_i + e_i$ (see Fig. 2.1). Notice that if $e_i = 0$, this fuzzy due-date becomes rigid (i.e., non-fuzzy), namely

$$\mu_{\boldsymbol{D}_i}(C_i) = \begin{cases} 1 & \text{if } C_i \leq d_i, \\ 0 & \text{if } C_i > d_i. \end{cases} \tag{2.2}$$

This means that we can represent a crisp due-date as a special case of fuzzy due-date.

Example 2.1

To show differences in DM's treatment of tardiness, we give an example of a crisp due-date $\boldsymbol{D}_1$ (of J_1) and two fuzzy due-dates $\boldsymbol{D}_2$ and $\boldsymbol{D}_3$ (of J_2 and J_3,

respectively) with the following membership functions (see Figs.2.2 and 2.3).

$$\mu_{\boldsymbol{D}_1}(C_1) = \begin{cases} 1 & \text{if } C_1 \leq 6, \\ 0 & \text{if } C_1 > 6, \end{cases}$$

$$\mu_{\boldsymbol{D}_2}(C_2) = \begin{cases} 1 & \text{if } C_2 \leq 10, \\ 11 - C_2 & \text{if } 10 < C_2 \leq 11, \\ 0 & \text{if } C_2 > 11, \end{cases}$$

$$\mu_{\boldsymbol{D}_3}(C_3) = \begin{cases} 1 & \text{if } C_3 \leq 3, \\ 1 - \dfrac{C_3 - 3}{6} & \text{if } 3 < C_3 \leq 9, \\ 0 & \text{if } C_3 > 9, \end{cases}$$

In this case, DM treats $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ as "rather strict" due-dates and $\boldsymbol{D}_3$ as a "non-strict" one. That is, he/she wants J_1 and J_2 completed by 6 ($= d_1$) and 10 ($= d_2$), respectively, where he allows that J_2 is completed shortly after 10 (e.g., if $C_2 = 10.1$, the satisfaction degree is 0.9) though he hopes that J_1 is completed by 6 if possible. On the other hand, he do not mind if J_3 is not completed by 3 ($= d_3$), but if the completion time is not less than 9 ($= d_3 + e_3$), he is dissatisfied with that. The value of e_3 (=6) is accordingly larger than e_1 (=0) and e_2 (=1), i.e., due-dates $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$ are more strict than $\boldsymbol{D}_3$. Thus, we can represent the differences in DM's treatment of tardiness job-wisely. ∎

2.2 Fuzzy Processing Times

Now let us introduce a fuzzy processing time as the second fuzzy factor which is a fuzzy number. For instance, such situations arise when we deal with persons (e.g., craftsmen) as machines or when machines are "unstable", e.g., due to some sudden rise or drop in temperature.

Let $\boldsymbol{P}_i$ $(i = 1, \cdots, n)$ be *fuzzy processing times* that are fuzzy numbers, for which we assume for simplicity that their membership functions $\mu_{\boldsymbol{P}_i}(x)$ are

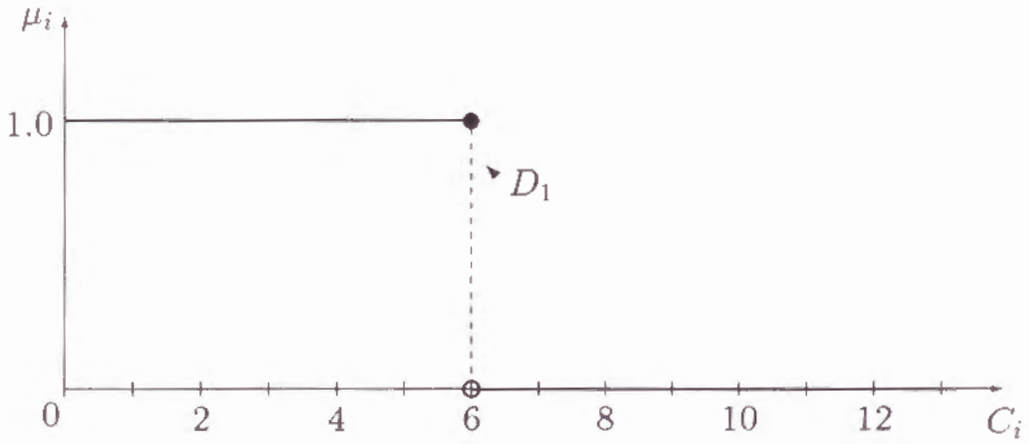

Fig. 2.2 Example of ordinary due-date

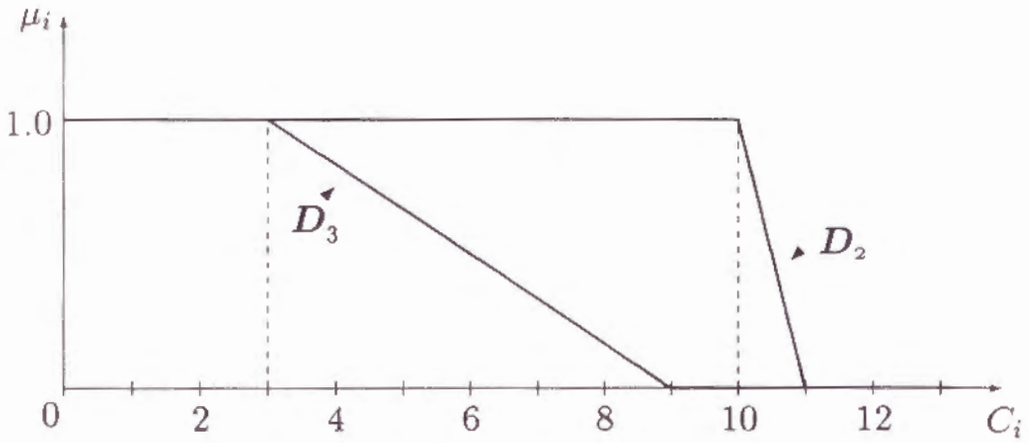

Fig. 2.3 Example of fuzzy due-dates

piecewise linear as shown in Fig.2.4, i.e.,

$$\mu_{\boldsymbol{P}_i}(x) = \begin{cases} \dfrac{x - p_i}{q_i} + 1 & \text{if } p_i - q_i \le x \le p_i, \\ \dfrac{p_i - x}{r_i} + 1 & \text{if } p_i < x \le p_i + r_i, \\ 0 & \text{otherwise,} \end{cases} \tag{2.3}$$

where p_i, q_i and r_i are positive constants. Equation (2.3) denotes that the most possible processing time of J_i is p_i.

Given some fuzzy processing times, we can calculate the completion time of all jobs from the definition of the addition of fuzzy numbers. Let us consider the following numerical example.

Example 2.2.

Assume that two fuzzy processing times $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are given as in Fig.2.5. Their membership functions $\mu_{\boldsymbol{P}_1}$ and $\mu_{\boldsymbol{P}_2}$ are defined as follows:

$$\mu_{\boldsymbol{P}_1}(x) = \begin{cases} x - 2 & \text{if } 2 \le x \le 3, \\ -x + 4 & \text{if } 3 < x \le 4, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\mu_{\boldsymbol{P}_2}(x) = \begin{cases} \dfrac{1}{2}x - \dfrac{5}{2} & \text{if } 5 \le x \le 7, \\ -x + 8 & \text{if } 7 < x \le 8, \\ 0 & \text{otherwise.} \end{cases}$$

From the definitions of $\mu_{\boldsymbol{P}_1}$ and $\mu_{\boldsymbol{P}_2}$, the intervals of confidence at level $\alpha \in [0, 1]$ are obtained as follows:

$$P_{1\alpha} = [\alpha + 2, -\alpha + 4]$$

and

$$P_{2\alpha} = [2\alpha + 5, -\alpha + 8].$$

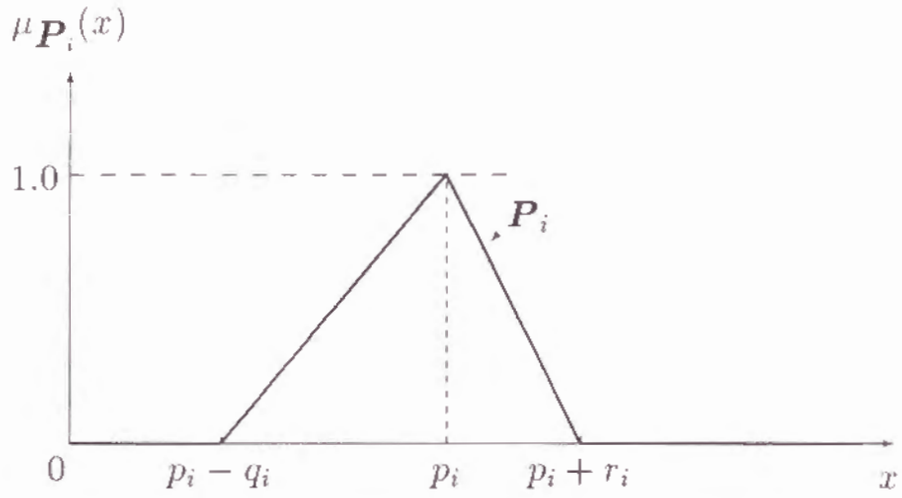

Fig. 2.4 Example of a fuzzy processing time

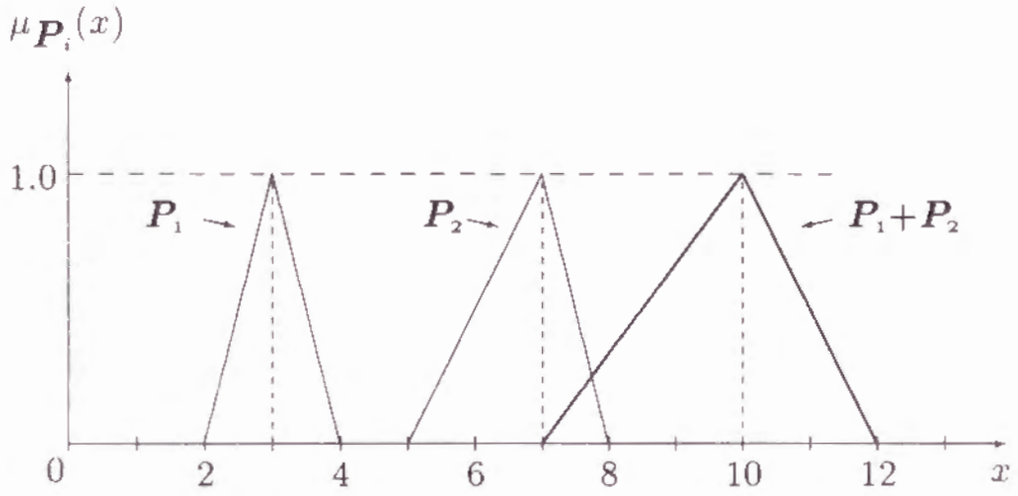

Fig. 2.5 Addition of fuzzy processing times

Then, the sum of these intervals is given by

$$P_{1\alpha} + P_{2\alpha} = [3\alpha + 7, -2\alpha + 12].$$

As a result, the following membership function, which shows the addition of these fuzzy processing times (i.e., the total processig time), is obtained.

$$\mu_{\boldsymbol{P}_1+\boldsymbol{P}_2}(x) = \begin{cases} \frac{1}{3}x - \frac{7}{3} & \text{if } 7 \le x \le 10, \\ -\frac{1}{2}x + 6 & \text{if } 10 < x \le 12, \\ 0 & \text{otherwise.} \end{cases}$$

If two jobs are processed in the order of $J_1 \rightarrow J_2$, this gives to the fuzzy completion time of J_2.

For the addition of fuzzy numbers whose membership functions are not linear, the similar procedure can be used. In this dissertation, however, we assume that fuzzy processing times are triangle shaped as shown in Figs. 2.4 and 2.5. We call this kind of fuzzy number a *triangular fuzzy number*. Since any triangular fuzzy number can be restored from the coordinates of three vertices of the triangle, we introduce the notation $TFN(x_l, x_c, x_r)$, where the subscripts l, c and r denote left, center and right points of the triangle (i.e., the corresponding coordinates are $(x_l, 0)$, $(x_c, 1)$ and $(x_r, 0)$, respectively).

For instance, the above fuzzy numbers $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ in Fig. 2.5 are expressed by $\boldsymbol{P}_1 = TFN(2,3,4)$, $\boldsymbol{P}_2 = TFN(5,7,8)$. Using this notation, the addition can be expressed more simply. That is, given n triangular fuzzy numbers (fuzzy processing times)

$$TFN(l_1, c_1, r_1), TFN(l_2, c_2, r_2), \cdots, TFN(l_n, c_n, r_n),$$

their sum (the total processing time) $\boldsymbol{T}_i$ is equal to

$$\boldsymbol{T}_i = TFN(\sum_{j=1}^{i} l_j, \sum_{j=1}^{i} c_j, \sum_{j=1}^{i} r_j). \tag{2.4}$$

Hence $\boldsymbol{T}_2 = TFN(2+5, 3+7, 4+8) = TFN(7, 10, 12)$. ∎

2.3 Fuzzy Precedence Constraints

As the third fuzzy factor, "fuzzy relation" can be applied to scheduling problems with fuzzy precedence constraints. The ordinary constraints are often represented by a *precedence graph* such that nodes and directed arcs correspond to jobs and their precedence, respectively (see Fig.2.6, where each number i in "node-circles" denotes J_i). If DM takes account of precedence constraints in view of various factors (e.g., some costs, technical constraints and more), it is not enough to express them by only ordinary relations. Accordingly *fuzzy precedence constraints* are introduced (i.e., we extend the graph to a fuzzy graph). The membership value associated with a relation between two jobs denotes DM's degree of satisfaction when these jobs are processed in the corresponding order. If a better schedule can be constructed by ignoring "non-strict" constraints, fuzzy precedence constraints allow their violations and the corresponding membership values are indicated.

Let jobs J_i and J_j have a fuzzy precedence relation $\boldsymbol{R}$ with its membership $\mu_{\boldsymbol{R}}(i,j) = 1$ (this is a simplified notation of $\mu_{\boldsymbol{R}}(J_i, J_j) = 1$) and $\mu_{\boldsymbol{R}}(j,i) = \alpha$, where α is a real number ($0 \leq \alpha < 1$). Here $\mu_{\boldsymbol{R}}(i,j) = 1$ means that the order $J_i < J_j$ is considered as normal, i.e., J_j is processed after the processing of J_i, and then DM is completely satisfied with the processing order. Otherwise (i.e., if the order is converse), his/her degree of satisfaction decreases from value 1 to α (< 1). Note that the symmetric law does not hold for fuzzy precedence relation introduced here (i.e., $\mu_{\boldsymbol{R}}(i,j) \neq \mu_{\boldsymbol{R}}(j,i)$). The resulting fuzzy relations defined on $\mathcal{J} \times \mathcal{J}$ are represented by a *fuzzy precedence matrix*, which is a "node-node incidence matrix" such that its (i,j)-th element denotes the value of $\mu_{\boldsymbol{R}}(i,j)$.

Example 2.3.

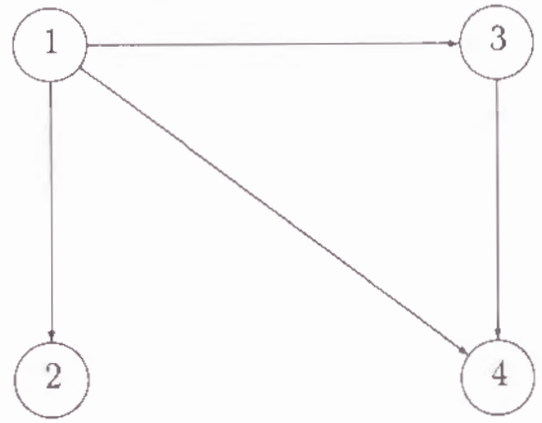

Fig. 2.6 Example of a precedence graph

We give a simple example of fuzzy precedence constraints for the above graph shown in Fig. 2.6, which consists of four "job-nodes" (we assume that $\mathcal{J} = \{J_1, \cdots, J_4\}$) and four "precedence-arcs" (i.e., $J_1 < J_2, J_1 < J_3, J_1 < J_4, J_3 < J_4$). Note that we implicitly assume that the directed graph is acyclic.

First, we set $\mu_{\boldsymbol{R}}(i,j) = 1$ for $J_i < J_j$, hence

$$\mu_{\boldsymbol{R}}(1,2) = \mu_{\boldsymbol{R}}(1,3) = \mu_{\boldsymbol{R}}(1,4) = \mu_{\boldsymbol{R}}(3,4) = 1.$$

Next, the degree of satisfaction for violation of each $J_i < J_j$ (i.e., in case J_j is processed before J_i) is determined by DM. If the original precedence relation $J_i < J_j$ is a "strict" (or "non-strict") one, the membership value for the violation $\mu_{\boldsymbol{R}}(j,i)$ is nearly equal to 0 (or 1). Otherwise (i.e., if he/she had better take account of the constraint "moderately"), the corresponding value becomes close to 0.5.

For instance, if he interprets fuzzy precedence relations $J_1 < J_3$ and $J_1 < J_4$

as strict constraints and $J_1 < J_2$ as a non-strict one, then a fuzzy precedence matrix may be given as follows:

$$\boldsymbol{R}: \begin{array}{c} \\ J_1 \\ J_2 \\ J_3 \\ J_4 \end{array} \begin{array}{c} \begin{array}{cccc} J_1 & J_2 & J_3 & J_4 \end{array} \\ \begin{bmatrix} \mathcal{I} & 1 & 1 & 1 \\ 0.9 & \mathcal{I} & \mathcal{I} & \mathcal{I} \\ 0.2 & \mathcal{I} & \mathcal{I} & 1 \\ 0.1 & \mathcal{I} & 0.4 & \mathcal{I} \end{bmatrix} \end{array}.$$

Here, an element indicated as "$\mathcal{I}$" denotes that there does not exist the constraint between the corresponding two jobs, i.e., they are independent (e.g., between J_2 and J_3). ∎

Chapter 3

ONE MACHINE PROBLEMS

3.1 One Machine Problem with Fuzzy Due-Dates

In this section, we consider three problems with fuzzy due-dates, i.e., problem $n \mid 1 \mid L_{\max}$ with fuzzy due-dates and two types of problem $n \mid 1 \mid L_{\max}$ with weighted fuzzy due-dates.

3.1.1 Problem with fuzzy due-dates

The fuzzy due-dates version of one machine problem ($n \mid 1 \mid L_{\max}$), maximizing the membership value of fuzzy due-dates is considered as a generalization of the original criterion (i.e., minimizing $L_{\max}$). The membership value $\mu_{\boldsymbol{D}_i}(C_i)$ defined by (2.1) shows DM's degree of satisfaction for completion time C_i. We call the resulting fuzzy scheduling problem $P1$: *fuzzy due-dates problem*. That is, the objective of $P1$ is to determine an optimal schedule that maximizes the smallest of all the membership values of fuzzy due-dates, i.e.,

$$P1: \quad \text{Maximize} \quad \min_i \mu_{\boldsymbol{D}_i}(C_i), \tag{3.1}$$

where

$$\mu_{\boldsymbol{D}_i}(C_i) = \begin{cases} 1 & \text{if } C_i < d_i, \\ 1 - \dfrac{C_i - d_i}{e_i} & \text{if } d_i \le C_i \le d_i + e_i, \\ 0 & \text{if } d_i + e_i < C_i. \end{cases} \tag{3.2}$$

Let a real number $t \in [0,1]$ be the minimum value of the objective function (3.1), i.e., $t = \min_i \mu_{\boldsymbol{D}_i}(C_i)$. To solve $P1$, it is sufficient to consider the range:

$$0 \le t \le 1, \tag{3.3}$$

that is, each C_i satisfies the following inequality:

$$C_i \le e_i(1-t) + d_i \ (\triangleq D_i(t)), \tag{3.4}$$

with the parametrized due-dates $D_i(t)$. Thus, $P1$ can be transformed into the one machine scheduling problem $P1'$ with the modified due-dates $D_i(t)$.

$$\begin{aligned} P1': \ & \underset{\pi \in \Pi}{\text{Maximize}} \quad t, && (3.5) \\ & \text{subject to} \\ & 0 \le t \le 1, \\ & C_{\pi(k)} = \sum_{j=1}^{k} p_{\pi(j)} \le d_{\pi(k)} + (1-t)e_{\pi(k)}, && (3.6) \\ & \qquad\qquad k = 1, 2, \cdots, n, \end{aligned}$$

where Π is a set of all sequences (permutations) of n jobs, and $\pi(j)$ denotes the j−th job in schedule $\pi \in \Pi$. If t is fixed, the problem is $n \mid 1 \mid L_{\max}$ and it is obvious that the following property is intutively appropriate from the *Earliest Due Date* (abbreviated to "*EDD*") rule [Smi].

Property 3.1 *An optimal solution of $P1$ is found by scheduling the jobs in non-decreasing order of the modified due-dates once t is fixed.*

Proof. Let $\mathcal{S}$ be any schedule and $\mathcal{S}^*$ a schedule determined according to the rule of Property 3.1. If $\mathcal{S} \neq \mathcal{S}^*$, there exist two jobs J_j and J_k with $D_k(t) \le D_j(t)$, such that J_j immediately precedes J_k in $\mathcal{S}$ but J_k precedes J_j in $\mathcal{S}^*$. Since $D_k(t) \le D_j(t)$, interchanging the positions of J_j and J_k in $\mathcal{S}$ cannot decrease the value of the objective function (3.5). A finite number of such changes transforms $\mathcal{S}$ into $\mathcal{S}^*$, showing that $\mathcal{S}^*$ is optimal. □

However, modified due-dates $D_i(t)$ change according to t, and we calculate all $t = t_{ij}$ satisfying $D_i(t) = D_j(t)$, i.e.,

$$t_{ij} = \frac{d_i - d_j + e_i - e_j}{e_i - e_j}, \quad i = 1, \cdots, n, j = 1, \cdots, n. \tag{3.7}$$

Then select only these t_{ij} that satisfy $0 < t < 1$, and sort such t_{ij} in the increasing order. The resulting sequence is denoted by

$$0 = t_0 < t_1 < \cdots < t_q = 1, \tag{3.8}$$

where $q-1$ is the number of different t_{ij} in (0,1), and hence $q - 1 \le n(n-1)/2$ holds. If DM's satisfaction degree is fixed to $t_c \in [t_{k-1}, t_k]$, an optimal processing order in this interval can be determined by the EDD-rule.

Fig.3.1 illustrates an example of a relation between membership functions $\mu_{\boldsymbol{D}_i}$ of jobs $J_i\,(i = 1,2,3)$ and their t_{ij}. In this case, the optimal order in each interval $[t_{k-1}, t_k], k = 1,2,3,4$ (where $t_0 = 0$, $t_1 = t_{23}$, $t_2 = t_{13}$, $t_3 = t_{12}$, $t_4 = 1$) is obtained as follows:

$$\begin{aligned} & J_1 \to J_2 \to J_3 && \text{for } t_c \in [t_3, t_4] \\ & J_2 \to J_1 \to J_3 && \text{for } t_c \in [t_2, t_3] \\ & J_2 \to J_3 \to J_1 && \text{for } t_c \in [t_1, t_2] \\ & J_3 \to J_2 \to J_1 && \text{for } t_c \in [t_0, t_1] \end{aligned}$$

However, not all of these schedules satisfy the condition of the modified due-dates (3.4). Accordingly, solving the problem is to find a maximum t_k such that

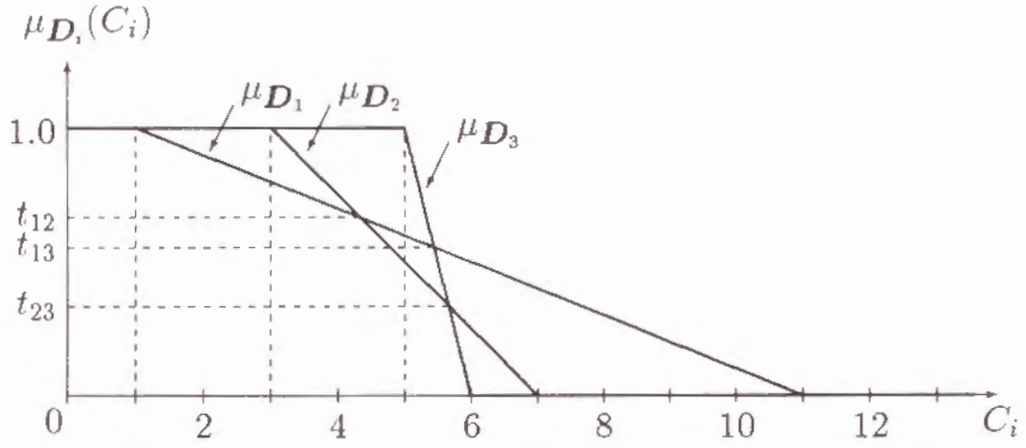

Fig. 3.1 Example of t_{ij}

the corresponding schedule satisfies condition (3.4). From the above fact, the following polynomial algorithm is proposed by using a binary search technique to the set of $\{t_0, t_1, \cdots, t_q\}$.

Algorithm 3.1

Step 0. After computing t_{ij} of (3.7) and sorting them to obtain (3.8), set $L := 0$ and $S := q$.

Step 1. If $S - L = 1$, then go to Step 4.

Step 2. After setting $A := \lfloor (L+S)/2 \rfloor$ (the notation "$\lfloor x \rfloor$" denotes the greatest integer not exceeding x) and $t_c := (t_A + t_{A+1})/2$, compute completion times C_i of jobs J_i scheduled in the increasing order of $D_i(t_c)$.

Step 3. If $C_i \leq D_i(t_A)$ holds for all i, then let the current processing order be

π, and after setting $L = A$, return to Step 1. Otherwise, after setting $S := A$, return to Step 1.

Step 4. We obtain the optimal value $t^* := \min_i \mu_{\boldsymbol{D}_i}(C_i)$, where $C_i = \sum_{j=1}^{i} p_{\pi(j)}$ ($\pi(j)$ denotes the $j-$th job in schedule π).

Theorem 3.1 *Algorithm 3.1 solves P1 in at most $O(n^2 \log n)$ computational time.*

Proof. The computational complexity for each step is as follows.

Step 0 : $O(n^2 \log n)$ to sort $O(n^2)$ t_{ij} 's.

Step 1 : $O(1)$.

Step 2 : $O(n \log n)$ to find a processing order of $O(n)$ J_i's and $O(n)$ to compute C_i's.

Step 3 : $O(n)$.

Step 4 : $O(n)$.

The loop of Step 1 to Step 3 is iterated at most $O(\log n)$ times because of the computation of binary search over $O(n^2)$ elements. Consequently, $P1$ can be solved in $O(n^2 \log n)$ computational time by Algorithm 3.1. □

Example 3.1

Now we consider a simple example of $P1$ in order to illustrate behavior of Algorithm 3.1. There are four jobs J_1 through J_4 whose processing times are given as $p_1 = 1, p_2 = 2, p_3 = 1$ and $p_4 = 3$. The membership functions that characterize fuzzy due-dates $\boldsymbol{D}_i, i = 1, 2, 3, 4$ are denoted by $\mu_i(C_i)$ in this example and they are given as follows (see Fig.3.2):

$$\mu_1(C_1) = \begin{cases} 1 & \text{if } C_1 < 3, \\ 1 - \dfrac{C_1 - 3}{5} & \text{if } 3 \leq C_1 \leq 8, \\ 0 & \text{if } 8 < C_1, \end{cases}$$

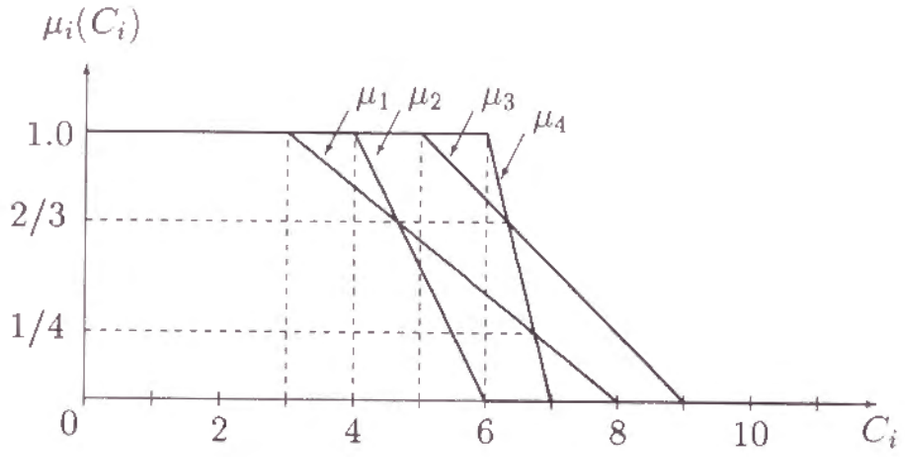

Fig. 3.2 Membership functions in Example 3.1

$$\mu_2(C_2) = \begin{cases} 1 & \text{if } C_2 < 4, \\ 1 - \dfrac{C_2 - 4}{2} & \text{if } 4 \le C_2 \le 6, \\ 0 & \text{if } 6 < C_2, \end{cases}$$

$$\mu_3(C_3) = \begin{cases} 1 & \text{if } C_3 < 5, \\ 1 - \dfrac{C_3 - 5}{4} & \text{if } 5 \le C_3 \le 9, \\ 0 & \text{if } 9 < C_3, \end{cases}$$

$$\mu_4(C_4) = \begin{cases} 1 & \text{if } C_4 < 6, \\ 1 - \dfrac{C_4 - 6}{1} & \text{if } 6 \le C_4 \le 7, \\ 0 & \text{if } 7 < C_4. \end{cases}$$

The 1-st iteration.

Step 0. The result of computing t_{ij} is as follows:

$$t_{12} = 2/3,\ t_{13} = -1,\ t_{14} = 1/4,\ t_{23} = 3/2,\ t_{24} = -1,\ t_{34} = 2/3.$$

After sorting them, we obtain

$$t_0 = 0 < t_1 = t_{14} = 1/4 < t_2 = t_{12} = t_{34} = 2/3 < t_3 = 1.$$

Set $L = 0$ and $S = 3$.

Step 1. Since $S - L = 3$, go to Step 2.

Step 2. Since $A = \lfloor 3/2 \rfloor = 1$ and $t_c = (t_1 + t_2)/2 = 11/24$, modified due-dates $D_i(t_c)$ are calculated as follows:

$$D_1(11/24) = 135/24,\ D_2(11/24) = 122/24,\ D_3(11/24) = 172/24,$$
$$D_4(11/24) = 157/24.$$

Accordingly, completion times C_i of jobs J_i for schedule $J_2 \to J_1 \to J_4 \to J_3$, are : $C_1 = 3, C_2 = 2, C_3 = 7$ and $C_4 = 6$.

Step 3. Since $C_i \le D_i(1/4)$ holds for all i, set $\pi = \{J_2 \to J_1 \to J_4 \to J_3\}$ and $L = 1$. Return to Step 1.

The 2-nd iteration.

Step 1. Since $S - L = 2$, go to Step 2.

Step 2. Since $A = \lfloor 4/2 \rfloor = 2$ and $t_c = (t_2 + t_3)/2 = 5/6$, modified due-dates $D_i(t_c)$ are calculated as follows:

$$D_1(5/6) = 23/6,\ D_2(5/6) = 26/6,\ D_3(5/6) = 34/6,\ D_4(5/6) = 37/6.$$

Accordingly, completion times C_i of jobs J_i for schedule $J_1 \to J_2 \to J_3 \to J_4$, are : $C_1 = 1, C_2 = 3, C_3 = 4$ and $C_4 = 7$.

Step 3. Since $C_i \le D_i(2/3)$ does not hold for $i = 4$, i.e., $C_4 = 7 \not\le D_4(2/3) = 19/3$, set $S = 2$ and return to Step 1.

The 3-rd iteration.

Step 1. Since $S - L = 1$, go to Step 4.

Step 4. We obtain the following optimal value:

$$t^* = \min\{\mu_1(3), \mu_2(2), \mu_3(7), \mu_4(6)\} = 0.5. \quad \blacksquare$$

3.1.2 Problems with weighted fuzzy due-dates

We can generalize the above problem $P1$ by introducing weights of jobs. A *weight* w_i is a positive real number, which represents the degree of importance of job J_i to DM. Based on w_i, we define *gain* $g_i(C_i)$ by

$$g_i(C_i) = w_i \mu_{\boldsymbol{D}_1}(C_i),\ i = 1, 2, \cdots, n. \tag{3.9}$$

We shall consider two problems $P2$ and $P3$. Problem $P2$ *maximizes min gains*, i.e.,

$$P2: \ \text{Maximize} \quad \left\{ \min_i g_i(C_i) \right\}. \tag{3.10}$$

Note that $P2$ is equivalent to $P1$ when we set all weights equal to 1. Comparing $P1$ and $P2$, it is not difficult to notice that $P2$ can be solved by replacing (3.4), (3.7) and (3.8) with the following (3.11), (3.12) and (3.13), respectively.

$$C_i \le e_i(1 - \frac{t}{w_i}) + d_i. \tag{3.11}$$

$$t_{ij} = \frac{d_i - d_j + e_i - e_j}{\frac{e_i}{d_i} - \frac{e_j}{w_j}},\ i = 1, \cdots, n,\ j = 1, \cdots, n, \tag{3.12}$$

$$0 = t_0 \le t_1 \le \cdots \le t_q = \min_i w_i. \tag{3.13}$$

Therefore this problem can also be solved in $O(n^2 \log n)$ time by simply considering the right hand side of (3.11) (i.e., $e_i(1 - \frac{t}{w_i}) + d_i$) as the new $D_i(t)$.

Problem $P3$ *maximizes total gain*, i.e.,

$$P3: \ \text{Maximize}_{\pi \in \Pi} \quad \sum_{j=1}^{n} g_{\pi(j)}(C_{\pi(j)}), \tag{3.14}$$

where

$$g_i(C_i) = w_i \mu'_{\boldsymbol{D}_1}(C_i), \tag{3.15}$$

$$\mu'_{\boldsymbol{D}_1}(C_i) = \begin{cases} 1 & \text{if } C_i < d, \\ 1 - \dfrac{C_i - d}{e_i} & \text{if } d \le C_i \le d + e_i, \\ 0 & \text{if } d + e_i < C_i, \end{cases} \tag{3.16}$$

$$C_{\pi(j)} = j. \tag{3.17}$$

Note that $\mu'_{\boldsymbol{D}_1}(C_i)$ differs from $\mu_{\boldsymbol{D}_1}(C_i)$ in that all d_i are the same value d, i.e., we assume that the original due-dates d_i of all jobs J_i are the same. We also assume that all processing times are unit ($p_i = 1$ for all i) and hence we have the equality of (3.17).

Theorem 3.2 *An optimal solution of $P3$ is found by scheduling jobs in a non-increasing order of w_i/e_i.*

Proof. We assume, without loss of generality, that $w_1/e_1 \ge w_2/e_2 \ge \cdots \ge w_n/e_n$ by rearranging job-indices if necessary. Consider the schedule $J_1 \to J_2 \to \cdots \to J_n$. Then, we show that the total gain does not increase even if the processing order of the $j-$th job and the $(j+1)-$st job are interchanged. Let $g_{j,j+1}$ be the gain obtained from processing the jobs J_j and J_{j+1} scheduled in this order and $g'_{j+1,j}$ the gain after interchanging the processing of these two jobs. We consider the following three cases.

(i) $C_{j+1}\ (= j+1) < d$

$$g_{j,j+1} = g'_{j+1,j} = w_j + w_{j+1}.$$

(ii) $C_j\,(=j)=d$

$$g_{j,j+1}=w_j+w_{j+1}-\frac{w_{j+1}}{e_{j+1}},\ g'_{j+1,j}=w_{j+1}+w_j-\frac{w_j}{e_j}.$$

Hence $g_{j,j+1}\geq g'_{j+1,j}$ (since $w_j/e_j\geq w_{j+1}/e_{j+1}$).

(iii) $C_j\,(=j)>d$

$$g_{j,j+1}=w_j+w_{j+1}-\frac{w_j(j-d)}{e_j}-\frac{w_{j+1}(j+1-d)}{e_{j+1}},$$

$$g'_{j+1,j}=w_{j+1}+w_j-\frac{w_{j+1}(j-d)}{e_{j+1}}-\frac{w_j(j+1-d)}{e_j}.$$

Hence $g_{j,j+1}\geq g'_{j+1,j}$ (since $g_{j,j+1}-g'_{j+1,j}=w_j/e_j-w_{j+1}/e_{j+1}\geq 0$). □

3.2 One Machine Problem with Fuzzy Processing Times

In this section, we consider a fuzzy processing time version of $(n\mid 1\mid L_{\max})$, where due-dates are not fuzzy but crisp. When processing times are fuzzy numbers, it is natural that their completion times becomes also fuzzy numbers. In order to formalize the problem, we use idea of "fuzzificated" crisp due-dates stated in Chapter 2, i.e., ordinary due-dates can be treated as special fuzzy due-dates as shown in Fig.2.2. Then, we calculate the agreement index of $\boldsymbol{C}_i$ with regard to $\boldsymbol{D}_i$ (see Definition 1.8), where $\boldsymbol{C}_i$ and $\boldsymbol{D}_i$ are fuzzy completion times and the "fuzzificated" crisp due-dates, respectively. The objective is to maximize the minimum of the agreement indices, i.e.,

$$P4:\ \text{Maximize}\quad \min_i\ \iota(\boldsymbol{C}_i,\boldsymbol{D}_i). \tag{3.18}$$

Now we assume that all processing times are so-called "quasi-similar" triangular fuzzy numbers for simplicity, i.e., when each processing time $\boldsymbol{P}_i$ is

represented by $TFN(l_i,c_i,r_i)$ (recall (2.4) in Section 2.2), and suppose that $1/\rho=(c_i-l_i)/(r_i-c_i)$ for all i, where ρ is a positive real number [Tad6,Ish4] (refer to [Bla] for another version of similar processing times). Given a processing order π, each completion time is also a "quasi-similar" triangular fuzzy number, i.e., the fuzzy completion time $\boldsymbol{C}_i$ is represented as

$$TFN(l'_i,c'_i,r'_i), \tag{3.19}$$

$$l'_i=\sum_{j=1}^{i}l_{\pi(j)},$$

$$c'_i=\sum_{j=1}^{i}c_{\pi(j)},$$

$$r'_i=\sum_{j=1}^{i}r_{\pi(j)},$$

where $\pi(j)$ denotes the $j-$th job in schedule π, and we have

$$\frac{1}{\rho}=\frac{c'_i-l'_i}{r'_i-c'_i}\left(=\frac{\sum_{j=1}^{i}c_{\pi(j)}-\sum_{j=1}^{i}l_{\pi(j)}}{\sum_{j=1}^{i}r_{\pi(j)}-\sum_{j=1}^{i}c_{\pi(j)}}\right) \tag{3.20}$$

We illustrate an example of fuzzy completion time $\boldsymbol{C}_i$ and fuzzificated due-date $\boldsymbol{D}_i$ in Fig.3.3.

Now we define further notations. Let $\mu_{\boldsymbol{C}_i}(x)$ be the membership function for fuzzy completion time, which shows a value of membership function when J_i is completed at time x (recall that both of the notations "$\boldsymbol{C}_i$" and "TFN" denote the fuzzy number of completion time, while the corresponding membership function is characterized by "$\mu_{\boldsymbol{C}_i}$" as in Example 2.2), and let

$$\alpha_i=\mu_{\boldsymbol{C}_i}(d_i), \tag{3.21}$$

(see Fig.3.3). Moreover let P_i^{α} (C_i^{α}) be an $\alpha-$cut of a fuzzy processing time $\boldsymbol{P}_i$ (fuzzy completion time $\boldsymbol{C}_i$) and $[P_{i1}^{(\alpha)},P_{i2}^{(\alpha)}]$ ($[C_{i1}^{(\alpha)},C_{i2}^{(\alpha)}]$) its interval of

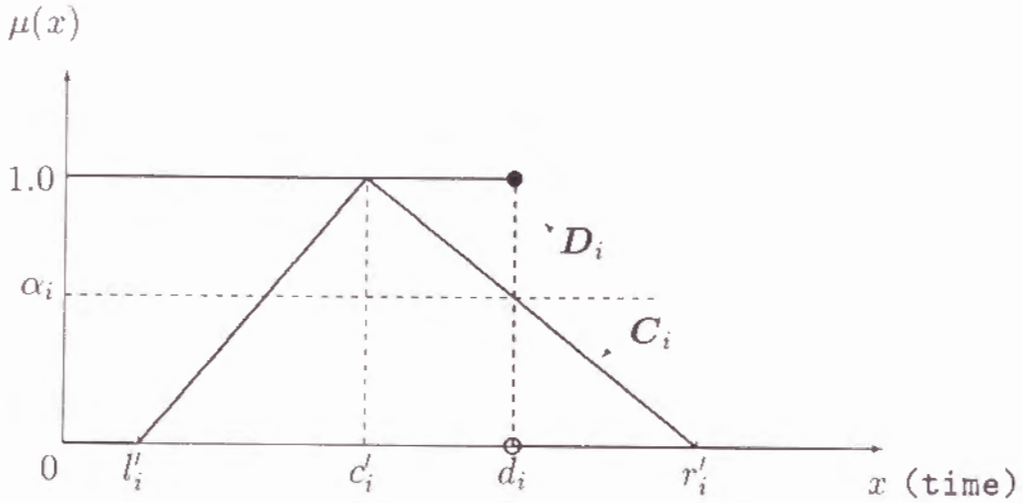

Fig. 3.3 Fuzzy processing time and fuzzificated due-date

confidence for the level of presumption α, i.e., $\mu_{\boldsymbol{P}_i}(P_{i1}^{(\alpha)}) = \mu_{\boldsymbol{P}_i}(P_{i2}^{(\alpha)}) = \alpha$ ($\mu_{\boldsymbol{C}_i}(C_{i1}^{(\alpha)}) = \mu_{\boldsymbol{P}_i}(C_{i2}^{(\alpha)}) = \alpha$).

Lemma 3.1 *Based on the above notations, the agreement index $\iota(\boldsymbol{C}_i, \boldsymbol{D}_i)$ is given by*

$$\iota(\boldsymbol{C}_i, \boldsymbol{D}_i) = \begin{cases} 1 - \dfrac{\rho\alpha_i^2}{\rho+1} & \text{if } d_i > c_i', \\ \dfrac{\alpha_i^2}{\rho+1} & \text{if } d_i \le c_i'. \end{cases}$$

Proof. Let $k = c_i' - l_i'$ (> 0). In case of $d_i > c_i'$, we have $r_i' - c_i' = k\rho$, $r_i' - l_i' = k(\rho+1)$ and $d_i - c_i' = k\rho - \alpha_i k\rho$, and hence

$$\begin{aligned} \iota(\boldsymbol{C}_i, \boldsymbol{D}_i) &= \frac{\dfrac{c_i' - l_i'}{2} + \dfrac{(d_i - c_i')(\alpha_i + 1)}{2}}{\dfrac{r_i' - l_i'}{2}} \\ &= \frac{1}{\rho+1} + \frac{k\rho(1-\alpha_i)(1+\alpha_i)}{k\rho + k} \end{aligned}$$

$$= 1 - \frac{\rho\alpha_i^2}{\rho+1}.$$

The other case is similar. □

Theorem 3.3 *There exists a schedule such that $\min_i \iota(\boldsymbol{C}_i, \boldsymbol{D}_i) \ge \kappa$, for a given κ such that $0 \le \kappa \le 1$, if and only if there exists a schedule such that the processing of every job $J_i, i = 1, 2, \cdots, n$ is completed by its due-date d_i when we regard $P_{i2}^{(\alpha)}, i = 1, 2, \cdots, n$ as their ordinary processing times if $\kappa > 1/(\rho+1)$ and $P_{i1}^{(\alpha)}$ otherwise, where the level α is given by*

$$\alpha = \begin{cases} \sqrt{(1-\kappa)(\rho+1)/\rho} & \text{if } \kappa > 1/(\rho+1), \\ \sqrt{\kappa(\rho+1)} & \text{if } \kappa \le 1/(\rho+1). \end{cases}$$

Proof. We show that the necessary and sufficient condition holds in both cases of

(I) $\kappa > 1/(\rho+1)$,

(II) $\kappa \le 1/(\rho+1)$.

Before the proof of this theorem, some notations used here are introduced and related properties are discussed.

First recall that the agreement index is calculated by either of the two right hand sides of the equation in Lemma 3.1, and denote the former and the latter (i.e., two cases of $d_i > c_i'$ and $d_i \le c_i'$) by case (i) and case (ii), respectively. Now assume that $\iota_q = \min_i \iota_i$ ($i = 1, \cdots, n$), where we let $\iota_i = \iota(\boldsymbol{C}_i, \boldsymbol{D}_i)$ for simplicity, namely, assume that the q−th job gives the minimum agreement index. Then, in case of (I), the agreement indices of all jobs are calculated by case (i) of Lemma 3.1 because $\iota_i \ge \iota_q > 1/(\rho+1)$ hold for all jobs J_i. In case (II), however, there may exist some jobs J_j satisfying $d_j > c_j'$ ($\iota_j > 1/(\rho+1)$) even if $d_q \le c_q'$ ($\iota_q \le 1/(\rho+1)$) holds. Therefore we need to check both the

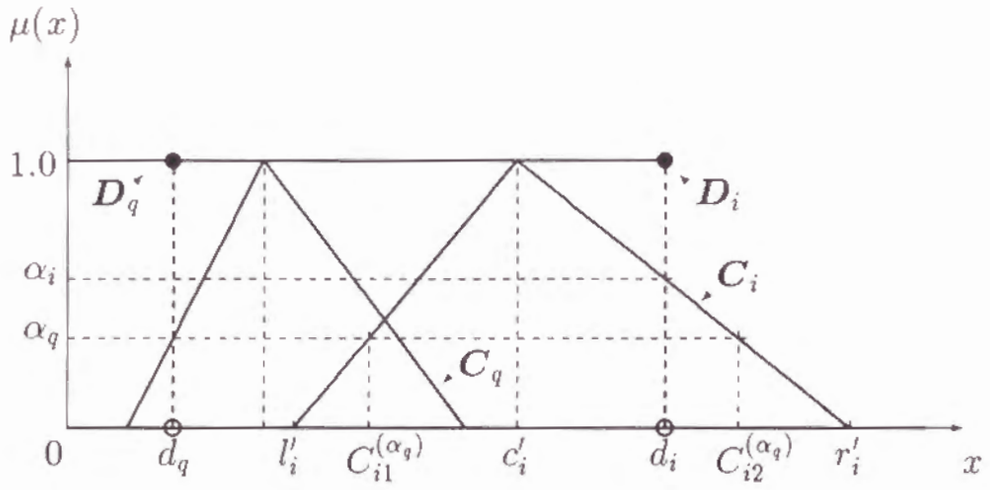

Fig. 3.4 Example of case (II-a)

situations (i) and (ii) for all jobs J_i for $i \neq q$ (see Fig.3.4), i.e., case (II) is divided into the following two sub-cases:

(II-a) There exists at least one job J_j such that $\iota_j > 1/(\rho+1)$.

(II-b) All jobs J_i satisfy that $\iota_i \leq 1/(\rho+1)$.

Next note that, in order to show the sufficient condition, we can assume without loss of generality that there exists a schedule satisfying $\iota_q \geq \kappa$ for a given κ, since any κ corresponds to either (I) or (II) (because $\kappa \in [0,1]$).

(Proof of I)

First assume that there exists a schedule satisfying $\min_i \iota_i = \iota_q \geq \kappa$, where $\kappa > 1/(\rho+1)$. From Lemma 3.1, we have

$$\iota_q = 1 - \frac{\rho\alpha_q^2}{\rho+1} \leq 1 - \frac{\rho\alpha_i^2}{\rho+1} = \iota_i,\ i = 1, \cdots, q-1, q+1, \cdots, n,$$

and it follows that $\alpha_q \geq \alpha_i$. Then we obtain that

$$\alpha_q = \mu_{\boldsymbol{C}_q}(C_{q2}^{(\alpha_q)}) = \mu_{\boldsymbol{C}_i}(C_{i2}^{(\alpha_q)}) \geq \mu_{\boldsymbol{C}_i}(d_i) = \alpha_i,$$

and hence we obtain $C_{i2}^{(\alpha_q)} \leq d_i$ (which is clear from the monotonicity of $\mu_{\boldsymbol{C}_i}(x)$ for $c_i' \leq x \leq r_i'$). Therefore the processing of every job is completed by its due-date when we regard $P_{i2}^{(\alpha)}$ as its ordinary processing time of job J_i, where $\alpha = \alpha_q = \sqrt{(1-\kappa)(\rho+1)/\rho}$ is obtained by solving the equation in case of (i) of Lemma 3.1 with $\iota_i = \kappa$ and $i = q$.

Conversely, if $C_{i2}^{(\alpha)} \leq d_i$ with $\alpha = \sqrt{(1-\kappa)(\rho+1)/\rho}$ holds for all i, we have

$$\mu_{\boldsymbol{C}_i}(C_{i2}^{(\alpha)}) = \sqrt{(1-\kappa)(\rho+1)/\rho} \geq \mu_{\boldsymbol{C}_i}(d_i) = \alpha_i,$$

and hence

$$\frac{1-\rho\alpha_i^2}{(\rho+1)} \geq \kappa$$

holds for all i, namely $\min_i \iota_i \geq \kappa$.

(Proof of II-a)

First assume that there exists a schedule satisfying $\min_i \iota_i = \iota_q \geq \kappa$, where $\kappa \leq 1/(\rho+1)$. We show that $C_j \leq d_j$ holds for every job J_j with $\iota_j > 1/(\rho+1)$, where C_j denotes a completion time of job J_j when we regard $P_{i1}^{(\alpha)}$ as its ordinary processing time for every job J_i (the proof for jobs other than such J_j is omitted here since it is obtained similarly to (II-b)). It is obvious that $c_j' < d_j$ and $C_{j1}^{(\alpha_q)} < c_j'$ hold for such j. Hence the processing of job J_j is completed by d_j when we regard $P_{i1}^{(\alpha)}$ as its ordinary processing time for every job J_i, where $C_i = \sum_{j=1}^{i} P_{j1}^{(\alpha)}$ holds for every job and the level α is obtained as $\alpha = \alpha_q = \sqrt{\kappa(\rho+1)}$ by solving the equation in case of (ii) of Lemma 3.1 with $\iota_i = \kappa$ and $i = q$.

Conversely, if there exists a schedule satisfying that $C_{i1}^{(\alpha)} \leq d_i$ with $\alpha = \sqrt{\kappa(\rho+1)}$ for all i when we regard $P_{i1}^{(\alpha)}$ as its ordinary processing time of J_i,

then we have

$$\iota_j = 1 - \frac{\rho\alpha^2}{\rho+1} \geq \iota_i = \frac{\alpha^2}{\rho+1} = \frac{\left\{\sqrt{\kappa(\rho+1)}\right\}^2}{\rho+1} = \kappa$$

from Lemma 3.1. Therefore $\iota_j \geq \kappa$ for all j.

(Proof of II-b)

First assume that there exists a schedule satisfying $\min_i \iota_i \geq \kappa$, $\kappa \leq 1/(\rho+1)$. From Lemma 3.1, we have

$$\iota_q = \frac{\alpha_q^2}{\rho+1} \leq \frac{\alpha_i^2}{\rho+1} = \iota_i,\ i = 1, \cdots, q-1, q+1, \cdots, n,$$

and it follows that $\alpha_q \leq \alpha_i$. Then we obtain that

$$\alpha_q = \mu_{\boldsymbol{C}_q}(C_{q1}^{(\alpha_q)}) = \mu_{\boldsymbol{C}_i}(C_{i1}^{(\alpha_q)}) \leq \mu_{\boldsymbol{C}_i}(d_i) = \alpha_i,$$

and hence we obtain $C_{i1}^{(\alpha_q)} \leq d_i$ (which is clear from the monotonicity of $\mu_{\boldsymbol{C}_i}(x)$ for $l_i' \leq x \leq c_i'$). Therefore the proessing of every job is completed by its due-date when we regard $P_{i1}^{(\alpha)}$ as its ordinary processing time of job J_i, where $\alpha = \alpha_q = \sqrt{\kappa(\rho+1)}$ is obtained by solving the equation in case of (ii) of Lemma 3.1 with $\iota_i = \kappa$ and $i = q$.

Conversely, if $C_{i1}^{(\alpha)} \leq d_i$ with $\alpha = \sqrt{\kappa(\rho+1)}$ holds for every i, we have

$$\mu_{\boldsymbol{C}_i}(C_{i1}^{(\alpha)}) = \sqrt{\kappa(\rho+1)} \leq \mu_{\boldsymbol{C}_i}(d_i) = \alpha_i,$$

and hence $\alpha_i^2/(\rho+1) \geq \kappa$ holds for all i, namely $\min_i \iota_i \geq \kappa$. □

From Theorem 3.3, assuming without loss of generality that job indices satisfiy the "EDD-order", i.e., $d_1 \leq d_2 \leq \cdots \leq d_n$, an optinal processing order of $P4$ is given by $J_1 \to J_2 \to \cdots \to J_n$.

3.3 One Machine Problem with Fuzzy Precedence Relations

In this section, we generalize one machine scheduling problem to minimize $L_{\max}$ (maximum lateness) under precedence constraints. That is, we consider the problem with fuzzy precedence relations. In this case, we have two objectives, i.e., $L_{\max}$ to be minimized and the minimum satisfaction level with respect to fuzzy precedence relation to be maximized.

3.3.1 Problem formulation

In order to formulate the problem, the following notations are used. Let $\pi(j)$ be the $j-$th job index of a schedule π. For simplicity, we denote by μ_{ij} $(=\mu_{\boldsymbol{R}}(J_i, J_j))$ the value of membership function of fuzzy precedence relation between two jobs J_i and J_j (in case J_i is processed before J_j). We assume that $\mu_{ji} = 1$ if $0 \leq \mu_{ij} < 1$, and both μ_{ij} and μ_{ji} equal 1 if jobs J_i and J_j are independent. In this section, however, we use notation $\mu_{ij} = \mu_{ji} = \mathcal{I}$ to denote that J_i and J_j are independent instead of $\mu_{ij} = \mu_{ji} = 1$, so that we can easily identify independent jobs. In this way, we can tell that if $\mu_{ji} = 1$ then there exists a fuzzy precedence relation such that $\mu_{ij} = \alpha$, $0 \leq \alpha < 1$. Further $L_{\max}^{\pi}$ is defined as the maximum lateness of a schedule π and

$$\mu_{\min}^{\pi} = \min\{\mu_{\pi(i)\pi(k)} \mid i, k = 1, 2, \cdots, n, i < k\}, \tag{3.22}$$

as the minimum value of membership μ_{ij}'s in π. Note that, in (3.22), we exclude all $\mu_{\pi(i)\pi(k)}$ with $i > k$. By using these notations, we consider the following bi-criteria scheduling problem $P5$.

$$P5: \quad L_{\max}^{\pi} \to \min,\ \mu_{\min}^{\pi} \to \max \tag{3.23}$$
$$\text{subject to } \pi \in \Pi,$$

where Π denotes the set of all feasible schedules. Generally speaking, there may not be a schedule that optimizes both the criteria $L^{\pi}_{\max}$ and $\mu^{\pi}_{\min}$ simultaneously. Accordingly, we introduce an idea of "nondominated schedule" (which is defined in the succeeding subsection).

3.3.2 Solving bi-criteria problem $P5$

First, before solving the bi-criteria problem, we briefly review a solution procedure of the (single criterion) scheduling problem that minimizes $L_{\max}$. Let

$$T_i = \{J_k \mid J_i < J_k, k \in \{1, 2, \cdots, n\}\}, \tag{3.24}$$

denote the set consisiting of jobs that J_i precedes. Lawler and Moore [Law2] showed that this problem is solved in $O(n^2)$ time by the assigning the following due-dates to job J_i,

$$d'_i = \min\{d_i, \min\{d_j \mid J_j \in T_i\}\}, \ i = 1, 2, \cdots, n. \tag{3.25}$$

and then by scheduling jobs in nondecreasing order of these due-dates, while observing precedence constraints. As a result, the original precedence relation is used to break "ties" (i.e., those jobs which have the same due-date).

Next we define nondominated schedules. In order to do that, we first introduce a *schedule vector* v^{π} consisting of two elements $L^{\pi}_{\max}$ and $\mu^{\pi}_{\min}$ of a schedule π:

$$v^{\pi} = (L^{\pi}_{\max}, \mu^{\pi}_{\min}). \tag{3.26}$$

For distinct schedule vectors $v^{\pi_1} = (v_1^{\pi_1}, v_2^{\pi_1})$ and $v^{\pi_2} = (v_1^{\pi_2}, v_2^{\pi_2})$, we call π_1 *dominates* π_2 (or π_1 is *more efficient* than π_2) if $v_1^{\pi_1} \leq v_1^{\pi_2}$ and $v_2^{\pi_1} \geq v_2^{\pi_2}$. If there exisits no schedule π' that dominates a schedule π, π is called *nondominated*. Since there exist, probably, many nondominated schedules

which have the same schedule vector, we concentrate our attention on finding at most one feasible schedule corresponding to each schedule vector.

Now let us discuss fuzzy precedence relation. Sort all different values among μ_{ij} into

$$\mu^0 \triangleq 1 > \mu^1 > \mu^2 > \cdots > \mu^k \geq 0, \tag{3.27}$$

where k is the number of different μ_{ij} (such that $\mu_{ij} \neq 1$). Note that k is at most $O(n^2)$. The following Lemma 3.2 is consequently clear.

Lemma 3.2 *Threre exist at most $O(n^2)$ nondominated schedule vectors in $P5$.*

An ordinary precedence relation is represented by the corresponding precedence graph $PG = [\mathcal{N}; \mathcal{A}]$ where $\mathcal{N}$ is the set of jobs $J_i, i = 1, 2, \cdots, n$ and $\mathcal{A}$ is the set of arcs (J_i, J_j) such that $J_i < J_j$, i.e., J_i precedes J_j. Recall that $\mu_{ij} = \mu_{ji} = \mathcal{I}$ for $i \neq j$ means that J_i and J_j are independent, i.e., none of $J_i < J_j$ and $J_j < J_i$ holds. In this case, naturally there exists none of (J_i, J_j) and (J_j, J_i) in PG. In our solution algorithm, we generalize the precedence graph for a fuzzy precedence relation, and use it in the dynamic manner in the sense that certain arcs are deleted after finding a nondeominated schedule at every iteration. As the initial graph, we use $PG^0 = [\mathcal{N}; A^0]$ which consists of the node set $\mathcal{N}$ and the arc set

$$A^0 \triangleq \{(J_i, J_j) \mid \mu_{ij} = \mu^0 \text{ and } \mu_{ji} \neq \mu^0\}, \tag{3.28}$$

where $\mu^0 = 1$. At the l−th iteration of the algorithm, $PG^l = [\mathcal{N}; A^l], l = 1, 2, \cdots, k$ (where $A^l \subset A^0$) are introduced, where A^l is the set of arcs defined by

$$\begin{aligned} &A^l = A^{l-1} - \overline{A}^l, \\ &\overline{A}^l \triangleq \{(J_i, J_j) \mid \mu_{ij} = \mu^0 \text{ and } \mu_{ji} = \mu^l\}, l = 1, 2, \cdots, k \end{aligned} \tag{3.29}$$

(i.e., $\overline{A}^l$ denotes the set of fuzzy relation arcs with level μ^l).

Note that the graph PG^l is not exactly the fuzzy graph defined in Definition 1.9 since we need only arcs (J_i, J_j) whose membership values μ_{ij} equal 1 in the algorithm. In other words, arcs (J_j, J_i) with $\mu_{ji} = \alpha$ $(0 \leq \alpha < 1)$ are not used when we find an optimal schedule in the sense of minimizing $L_{\max}$ at every iteration in the algorithm. That is, once the precedence graph PG^l is determined, the problem is reduced to the ordinary problem that minimizes the maximum lateness with precedence relations represented by the crisp graph. Note also that finding a schedule minimizing $L_{\max}$ from the graph PG^l means that we take no account of precedence relations $J_i < J_j$ corresponding to $\mu_{ji} \in [\mu^l, \mu^0]$ at the $l-$th iteration. Further notations used in the algorithm are introduced, i.e., two variables DV and DS are prepared to store nondominated schedule vectors and the corresponding schedules, respectively.

Now we are ready to describe our solution procedure in order to find nondominated schedules of $P5$. As mentioned above, we try to find at most one nondominated schedule corresponding to each scheduling vector, whose second element is μ^l $(l = 0, 1, \cdots, k)$, and hence we find at most $(k+1)$ nondominated schedules. That is, the number of nondominated scheduling vectors does not always correspond to that of μ^l's (namely, $k+1$), since there may exist a nondominated schedule π such that the resulting value $L^{\pi}_{\max}$ is obtained without violating precedence relations $J_i < J_j$ with $\mu_{ji} < \mu^{\pi}_{\min}$ (in this case, even if arcs (i, j) are excluded from the graph at a certain iteration, it does not improve the value of $L_{\max}$; for details, see the proof of Theorem 3.4).

Algorithm 3.2

Step 0. Solve one machine $L_{\max}$ minimization problem with precedence relations $PG^0 = [\mathcal{N}; A^0]$. Let its optimal maximum lateness value be

$L^0_{\max}$ and the corresponding optimal schedule be π^0. Set $DV := \{(L^0_{\max}, 1)\}$, $DS := (\pi^0)$ and $l := 1$.

Step 1. After setting $A^l := A^{l-1} - \overline{A}^l$ and constructing $PG^l = [\mathcal{N}; A^l]$, solve one machine $L_{\max}$ minimization problem with precedence relation $PG^l = [\mathcal{N}; A^l]$. Let its optimal value of $L_{\max}$ be $L^l_{\max}$ and the corresponding schedule be π^l. Construct the corresponding schedule vector $v^l := (L^l_{\max}, \mu^l_{\min})$ where $\mu^l_{\min} = \min\{\mu_{\pi^l(i)\pi^l(j)} \mid i, j = 1, 2, \cdots, n, i < j\}$. If v^l is dominated by some vector in DV or already included in DV, then go to Step 2. Otherwise, after setting $DV := DV \cup \{v^l\}$ and $DS := DS \cup \{\pi^l\}$, go to Step 2.

Step 2. Set $l := l + 1$. If $l = k + 1$, terminate. Otherwise, return to Step 1.

Theorem 3.4 *Algorithm 3.2 calculates the set of all nondominated schedules of P5 as final DS, and each of the resulting schedules corresponds to exactly one nondominated schedule vector. Complexity of the algorithm is $O(n^4)$.*

Proof. Clearly the second element of each nondominated scheduling vector is one of $\mu^l, l = 0, 1, 2, \cdots, k$. Note that the deletion of arcs in $\overline{A}^l$ at Step 1 corresponds to the exclusion of both precedences $J_i < J_j$ and $J_i > J_j$ such that $\mu_{ij} = \mu^0$ and $\mu_{ji} = \mu^l$. After this, J_i and J_j become mutually independent, if there exists none of arcs incident to J_i and J_j, e.g., none of arcs $(J_i, J_{j'})$ and $(J_{j'}, J_j)$, where $J_{j'}$ is a job such that $J_i < J_{j'}$ and $J_{j'} < J_j$. At the last $((k+1)$-st) iteration, all jobs consequently become independent.

Now we assume that the $(l-1)$-st iteration has just finished and assume, without loss of generality, that all the jobs J_i and J_j such that arcs $(J_i, J_j) \in \overline{A}^l$ become independent after the deletion of these arcs. Let l' be the greatest index of nondominated vectors in the current DV. At the l-th iteration, we

obtain the nondominated schedule with $(L^l_{\max}, \mu^l_{\min})$ such that $L^l_{\max} < L^{l'}_{\max}$ and $\mu^l_{\min} < \mu^{l'}_{\min}$, if there exists no jobs J_i and J_j such that the due-date of J_j is earlier than that of J_i, i.e., $d_i > d_j$. Otherwise (namely, if $d_i \leq d_j$), the exclusion of precedence relation $J_i < J_j$ at the l-th iteration does not improve the value of $L_{\max}$, since the nondominated schedule obtained at the l'-th iteration is the same as the schedule obtained by solving the problem of minimizing $L_{\max}$ with the remaining precedence relations represented by PG^l. That is, if $d_i \leq d_j$ for the jobs J_i and J_j then the resulting schedule at the l'-th iteration naturally preserves the constraints $J_i < J_j$ and hence that at the l-th iteration is not nondominated. The algorithm searches nondominated schedule vector by changing the second element from μ^0 till μ^k. Thus, the algorithm calcultes the set of all nondomnated schedules, each of which corresponds to exactly one nondominated schedule vector.

While, complexity of the algorithm is k $(= O(n^2))$ times of the complexity of the solution algorithm for solving one machine $L_{\max}$ minimization problem with precedence relations. Recall that the complexity of the latter is $O(n^2)$. Consequently the total complexity is $O(n^4)$. □

Example 3.2

The following illustrates a behavior of Algorithm 3.2. There are five jobs with the fuzzy precedence relation shown in Tab.3.1, where $\mathcal{I}$ means corresponding pairs of jobs are mutually independent (e.g., jobs J_2 and J_3 are independent). The corresponding precedence graph $PG^0 = [\mathcal{N}; A^0]$ is shown in Fig.3.5. The processing time and due-date of each job are shown in Tab.3.2. Before executing the algorithm, sort all μ_{ij} to obtain

$$\mu^0 = 1 > \mu^1 = 0.8 > \mu^2 = 0.6 > \mu^3 = 0.5 > \mu^4 = 0.2 > 0.$$

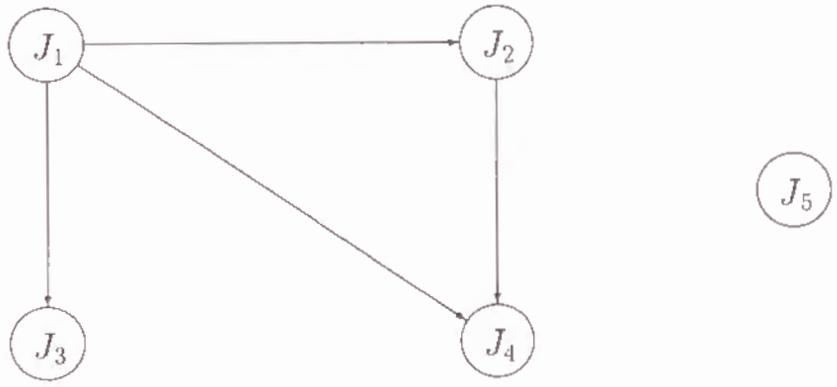

Fig. 3.5 Precedence graph $PG^0 = [\mathcal{N}; A^0]$

Tab. 3.1 Fuzzy precedence relation

	J_1	J_2	J_3	J_4	J_5
J_1	–	1	1	1	$\mathcal{I}$
J_2	0.5	–	$\mathcal{I}$	1	$\mathcal{I}$
J_3	0.2	$\mathcal{I}$	–	$\mathcal{I}$	$\mathcal{I}$
J_4	0.8	0.6	$\mathcal{I}$	–	$\mathcal{I}$
J_5	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$	$\mathcal{I}$	–

Tab. 3.2 Processing time and due-date of each job

Job	J_j :	J_1	J_2	J_3	J_4	J_5
Processing time	p_j :	2	1	3	1	2
Due-date	d_j :	8	9	5	2	6

The 1-st iteration.

Step 0. Since modified due-dates d'_j of (3.25) are

$$(d'_1, \cdots, d'_5) = (2, 2, 5, 2, 6),$$

the processing order of schedule π^0 and the resulting lateness are shown in Tab.3.3. Hence we have

$$DV = \{(3,1)\} \ \left(= (L^0_{\max}, \mu^0_{\min})\right),$$
$$DS = \left\{\pi^0\right\} \ (= \{J_1 \to J_2 \to J_4 \to J_3 \to J_5\}) \text{ and } l = 1.$$

Step 1. Since $\overline{A}^1 = \{(J_1, J_4)\}$, $PG^1 = [\mathcal{N}; A^1]$ becomes as shown in Fig.3.6. Modified due-dates are

$$(d'_1, \cdots, d'_5) = (2, 2, 5, 2, 6).$$

Note that $d'_1 = 2$ since $T_1 = \{J_2, J_3, J_4\}$ after deleting arc (J_1, J_4). Then, schedule π^1 is

$$\pi^1 : \ J_1 \to J_2 \to J_4 \to J_3 \to J_5.$$

Then Step 2 is immediately entered since this processing order is the same as π^0.

Step 2. Set $l = 2$ and return to Step 1.

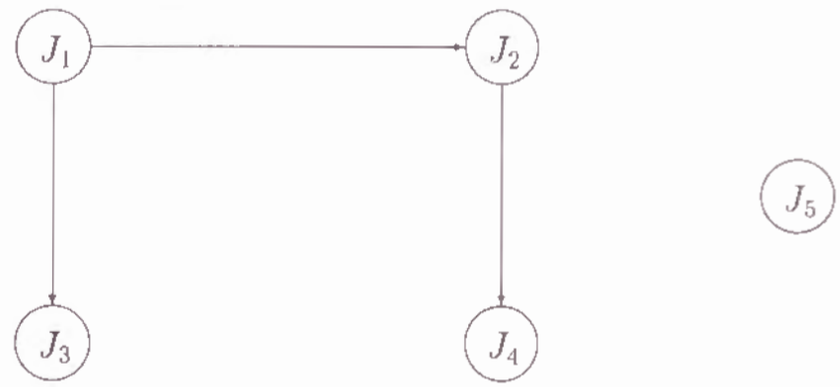

Fig. 3.6 Precedence graph $PG^1 = [\mathcal{N}; A^1]$

The 2-nd iteration.

Step 1. Since $\overline{A}^2 = \{(J_2, J_4)\}$, $PG^2 = [\mathcal{N}; A^2]$ is constructed as shown in Fig.3.7. Then modified due-dates are

$$(d'_1, \cdots, d'_5) = (5, 9, 5, 2, 6).$$

The processing order of schedule π^2 and the resulting lateness are shown in Tab.3.4. Since we have $v^2 = (2, 0.6)$, which is nondomi-

Tab. 3.3 Results from schedule π^1

Processing order	π^0 :	$J_1 \longrightarrow$	$J_2 \longrightarrow$	$J_4 \longrightarrow$	$J_3 \longrightarrow$	J_5
Processing time	p_j :	2	1	1	3	2
Completion time	C_j :	2	3	4	7	9
Due-date	d_j :	8	9	2	5	6
Lateness	L_j :	-6	-6	2	2	3

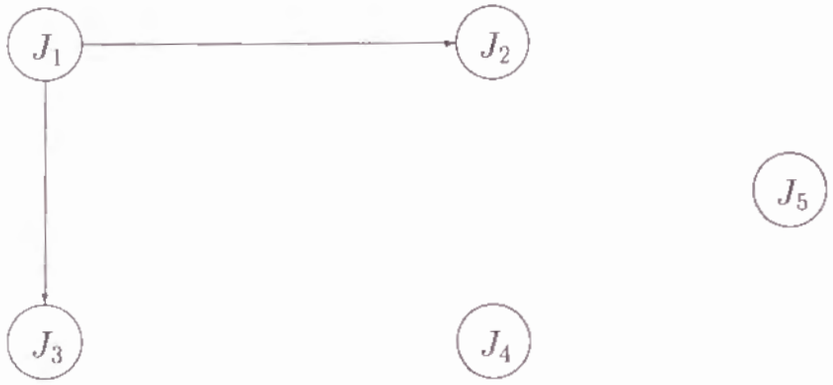

Fig. 3.7 Precedence graph $PG^2 = [\mathcal{N}; A^2]$

nated with respect to the current DV, the new DV and DS become

$$DV = \{(3,1)\} \cup \{(2,0.6)\}\,,\ DS = \left\{\pi^0\right\} \cup \left\{\pi^2\right\}.$$

Then go to Step 2.

Step 2. Set $l = 3$ and return to Step 1.

The 3-rd iteration.

Tab. 3.4 Results from schedule π^2

Processing order	π^2:	$J_4 \longrightarrow$	$J_1 \longrightarrow$	$J_3 \longrightarrow$	$J_5 \longrightarrow$	J_2
Processing time	p_j:	1	2	3	2	1
Completion time	C_j:	1	3	6	8	9
Due-date	d_j:	2	8	5	6	9
Lateness	L_j:	-1	-5	1	2	0

Fig. 3.8 Precedence graph $PG^3 = [\mathcal{N}; A^3]$

Step 1. Since $\overline{A}^3 = \{(J_1, J_2)\}$, $PG^3 = [\mathcal{N}; A^3]$ becomes as shown in Fig.3.8, and modified due-dates are

$$(d_1', \cdots, d_5') = (5, 9, 5, 2, 6),$$

which are the same as previous ones. Consequently, we go to Step 2 immediately.

Step 2. Set $l = 4$ and return to Step 1.

The 4-th iteration.

Step 1. Since $\overline{A}^4 = \{(J_1, J_3)\}$, $PG^4 = [\mathcal{N}; A^4]$ contains no arc, i.e., all jobs become independent. Then modified due-dates becomes the original due-dates not taking into account the precedence relatrions, i.e.,

$$(d_1', \cdots, d_5') = (8, 9, 5, 2, 6).$$

The processing order π^4 and the resulting lateness are shown in

Tab. 3.5 Results from schedule π^4

Processing order	π^4:	$J_4 \longrightarrow$	$J_3 \longrightarrow$	$J_5 \longrightarrow$	$J_1 \longrightarrow$	J_2
Processing time	p_j:	1	3	2	2	1
Completion time	C_j:	1	4	6	8	9
Due-date	d_j:	2	5	6	8	9
Lateness	L_j:	-1	-1	0	0	0

Tab.3.5. Since $v^4 = (0, 0.2)$ is nondominated, set

$$DV = \{(3,1),(2,0.6)\} \cup \{(0,0.2)\}\,,\ DS = \left\{\pi^0, \pi^2\right\} \cup \left\{\pi^4\right\},$$

and go to Step 2.

Step 2. Since $l = 5$, the algorithm terminates.

Thus, we obtain three nondominated schedules π^0, π^2 and π^4. ∎

Chapter 4

MULTI MACHINE PROBLEMS

4.1 Identical Machine Problem with Fuzzy Due-dates

4.1.1 Generalized $n \mid m \mid I \mid L_{\max}$ problem

Before cosidering a fuzzy version of identical machines scheduling problem, we consider a generalized version of the original one, i.e., $n \mid m \mid I \mid L_{\max}$ preemptive scheduling problem with general due-dates (for specification of the problem and definitions of these terms; refer to Section 1.3). We assume that all processing times p_i and due-dates d_{ij} are non-negative integers. This problem can be solved by finding a minimum value of L such that there exists a *feasible* schedule which processing of each job J_i is completed by the following *modified due-dates* d'_{ij}:

$$d'_{ij} = d_{ij} + L, \quad i = 1, 2, \cdots, n, j = 1, 2, \cdots, m. \tag{4.1}$$

Then we determine whether a schedule is feasible or not by using a reduced network $G = [\mathcal{N}; \mathcal{A}]$ defined in the following (II), and utilize a binary search technique to find the minimum L.

Now we propose a solution procedure for this problem.

Setting of initial values

(I) Sort all d'_{ij} in increasing order, i.e.,

$$D_0 \triangleq 0 < D_1 < D_2 < \cdots < D_k,$$

where k is the number of different due-dates, and D_h denotes the h-th smallest value among d'_{ij}'s.

Then we introduce interval $I_h = [D_{h-1}, D_h]$ and

$$s_h = D_h - D_{h-1}, \quad h = 1, 2, \cdots, k. \tag{4.2}$$

(II) Construct the reduced network $G = [\mathcal{N}; \mathcal{A}]$ (see Fig.4.1) such that

$$\mathcal{N} = N_1 \cup N_2 \cup N_3 \cup N_4,$$

where

$$N_1 = \{M_{jh} \mid j = 1, \cdots, m, h = 1, \cdots, k\}, \tag{4.3}$$

and each element M_{jh} of N_1 corresponds to machine M_j in the interval I_h ($\mid N_1 \mid = km$),

$$N_2 = \{V_{hi} \mid h = 1, \cdots, k, i = 1, \cdots, n\}, \tag{4.4}$$

and each element V_{hi} of N_2 corresponds to job J_i in the interval I_h ($\mid N_2 \mid = kn$),

$$N_3 = \{J_i \mid i = 1, \cdots, n\}, \tag{4.5}$$

and each element of N_3 is job J_i ($\mid N_3 \mid = n$), and

$$N_4 = \{s, t\}$$

representing source and sink, respectively ($\mid N_4 \mid = 2$). Moreover let

$$\mathcal{A} = A_1 \cup A_2 \cup A_3 \cup A_4,$$

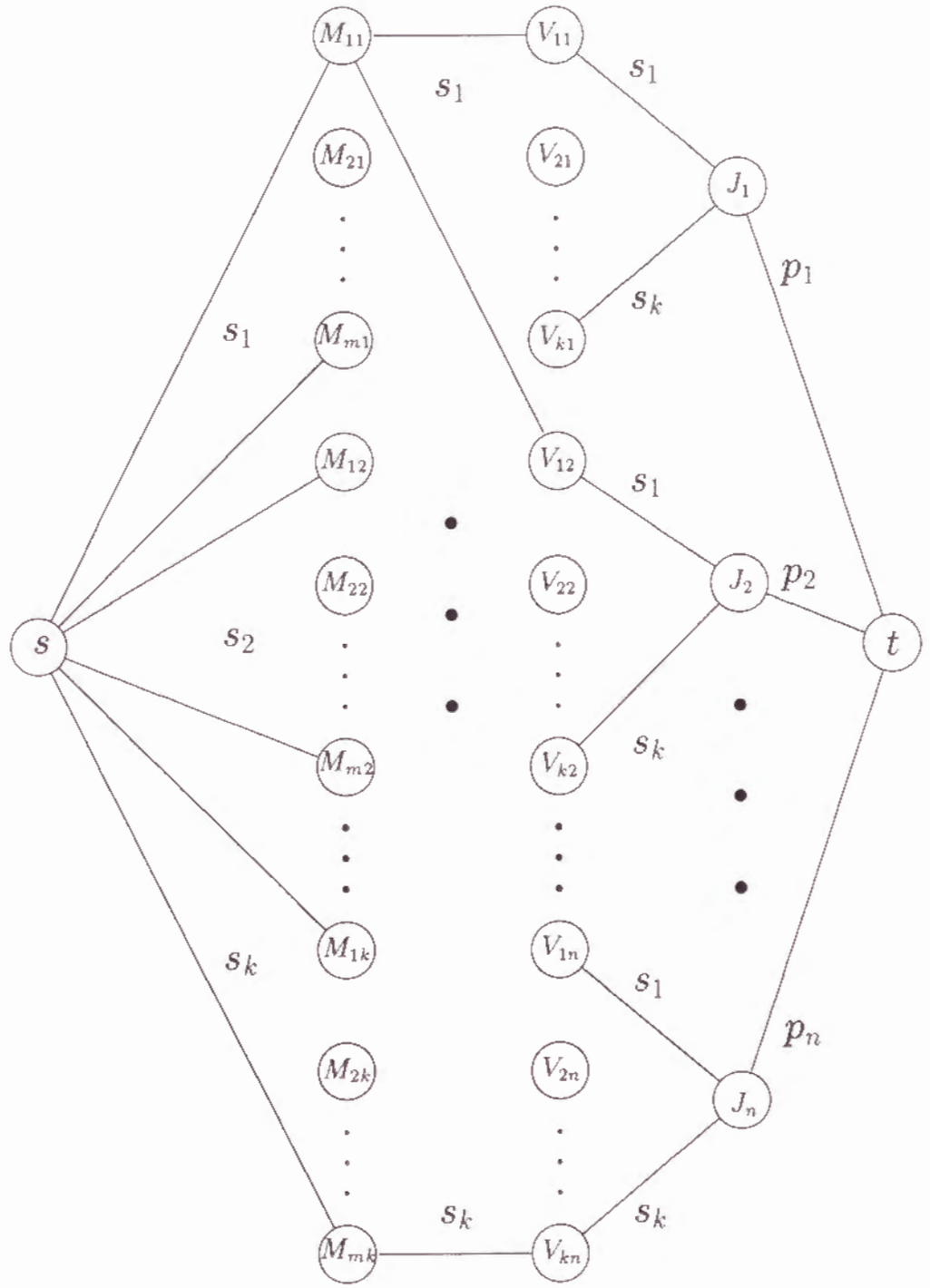

Fig.4.1 Example of reduced network

where

$$A_1 = \{(s, M_{jh}) \mid j = 1, \cdots, m, h = 1, \cdots, k\} \tag{4.6}$$

and capacity of each arc (s, M_{jh}) is s_h,

$$A_2 = \{(J_i, t) \mid i = 1, \cdots, n\} \tag{4.7}$$

and capacity of each arc (J_i, t) is p_i,

$$A_3 = \{(V_{hi}, J_i) \mid h = 1, \cdots, k, i = 1, \cdots, n\} \tag{4.8}$$

and capacity of each arc (V_{hi}, J_i) is s_h, and

$$A_4 = \{(M_{jh}, V_{hi}) \mid d'_{ij} \geq D_h, h = 1, \cdots, k, i = 1, \cdots, n, j = 1, \cdots, m\} \tag{4.9}$$

and capacity of each arc (M_{jh}, V_{hi}) is s_h.

Note that each arc capacity of the reduced network has the following meaning.

i. Capacity s_h of each arc $(S, M_{jh}) \in A_1$: maximum available time of machine M_j for processing jobs in interval I_h.

ii. Capacity p_i of each arc $(J_i, T) \in A_2$: processing time of job J_i.

iii. Capacity s_h of each arc $(V_{hi}, J_i) \in A_3$: maximum allowable processing time for job J_i in interval I_h.

iv. Capacity s_h of each arc $(M_{jh}, V_{hi}) \in A_4$: maximum allowable assignment of processing time of job J_i on machine M_j in interval I_h.

In addition, the arc flow of $(M_{jh}, V_{hi}) \in A_4$ corresponds to the processing time assignment of job J_i to machine M_j in interval I_h, and the condition $d'_{ij} \geq D_h$ means that the interval not later than the modified due-date of

job J_i is available for processing J_i. Hence there exists a feasible schedule that completes all jobs J_i on machine M_j by d'_{ij} if and only if there exists, for the reduced network, a maximal flow with flow value $\sum_{i=1}^{n} p_i$, since the corresponding preemptive open shop problem, i.e., $n \mid m \mid O \mid C_{\max}$ assures that the transformation of flow $f(M_{jh}, V_{hi})$ into the actual processing time assignment of J_i to M_j in interval I_h is possible (for details; see [Gon]). Thus the feasibility check and the construction of a feasible schedule can be executed.

Now we are ready to describe the algorithm for solving the problem $n \mid m \mid I \mid L_{\max}$ preemptive scheduling problem with general due-dates (validity of the algorithm is shown as a natural extension of [Mas] and so it is omitted here).

Algorithm 4.1

Step 0. Compute $L := \lfloor (L^{(1)} + L^{(2)})/2 \rfloor$, where $L^{(1)} = 0, L^{(2)} = \sum_{i=1}^{n} p_i$.

Step 1. Set $d'_{ij} := d_{ij} + L$ and construct the reduced network $G = [\mathcal{N}; \mathcal{A}]$ in the above manner.

Step 2. Find a maximum flow of G and let its flow value be f, and then

(a) if $f = \sum_{i=1}^{n} p_i$, go to Step 3 (there exists a feasible schedule),

(b) otherwise (i.e., if $f < \sum_{i=1}^{n} p_i$), go to Step 4 (there does not exist a feasible schedule).

Step 3. Let $L^{(2)} := L$. If $L^{(1)} = L^{(2)}$ then go to Step 5. Otherwise set $L := \lfloor (L^{(1)} + L^{(2)})/2 \rfloor$ and return to Step 1.

Step 4. Let $L^{(1)} := L$. If $L^{(1)} = L^{(2)}$ then go to Step 5. Otherwise set $L := \lfloor (L^{(1)} + L^{(2)})/2 \rfloor$ and return to Step 1.

Step 5. Construct the preemptive scheduling problem $n \mid m \mid O \mid C_{\max}$ with the following processing time p_{ij}^h (of job J_i on machine M_j) corresponding to each interval $I_h, h = 1, 2, \cdots, k$. If there exists arc (M_{jh}, V_{hi}), then set $p_{ij}^h = f(M_{jh}, V_{hi})$. Otherwise, $p_{ij}^h = 0$. Then solve the open shop problem with the processing time p_{ij}^h for each interval I_h by using well-known Gonzalez and Shani's algorithm [Gon] in order to construct partial schedules and concatenate these schedules to construct a feasible total schedule.

4.1.2 Fuzzy general due-dates version

We consider a fuzzy general due-dates version of the problem $n \mid m \mid I \mid L_{\max}$ described in Subsection 4.1.1. The following membership function $\mu_{ij}(C_{ij})$ is introduced to denote DM's satisfaction with respect to completion time C_{ij} of each job J_i in case that J_i is finally completed on M_j.

$$\mu_{ij}(C_{ij}) = \begin{cases} 1 & \text{if } C_{ij} \le d_{ij}, \\ 1 - \dfrac{C_{ij} - d_{ij}}{e_{ij}} & \text{if } d_{ij} < C_{ij} \le d_{ij} + e_{ij}, \\ 0 & \text{if } C_{ij} > d_{ij} + e_{ij}, \end{cases} \tag{4.10}$$

where d_{ij} and e_{ij} are nonnegative integers. The objective is to find a schedule that maximizes the minimum satisfaction among all $\mu_{ij}(C_{ij})$, that is,

$$P6: \quad \text{Maximize} \quad \min_{i,j} \mu_{ij}(C_{ij}), \quad i = 1, 2, \cdots, n, \; j = 1, 2, \cdots, m. \tag{4.11}$$

Note that if all values of $e_{ij} - d_{ij}$ are the same, the problem reduces to the ordinary problem $n \mid m \mid I \mid L_{\max}$ with general due-dates.

The solution procedure is described as follows. First we check the feasibility of modified due-dates set as $d'_{ij} = d_{ij}$ (i.e., $L = 0$) by using of the reduced network similarly to Algorithm 4.1. If it is feasible, we obtain an optimal schedule such that the value of objective function (4.11) is equal to 1. Otherwise, we

calculate satisfaction values $t_{(i,j),(h,l)}$'s at intersection points of the functions (4.10), that is,

$$t_{(i,j),(h,l)} = \mu_{ij}(C_{ij}) = \mu_{hl}(C_{ij}) \tag{4.12}$$

for all pairs $((i,j),(h,l))$. Sort them in increasing order, i.e.,

$$t_0 \triangleq 0 < t_1 < \cdots < t_r < 1 \triangleq t_{r+1}, \tag{4.13}$$

where r is the number of different $t_{(i,j)(h,l)}$'s and t_h is the h-th smallest value among $t_{(i,j)(h,l)}$'s. We call a satisfaction level t *feasible* when there exists a schedule that completes all jobs by the following modified due-dates:

$$d'_{ij} = \mu_{ij}^{-1}(t), \quad i = 1, \cdots, n, j = 1, \cdots, m. \tag{4.14}$$

In this case, the maximum feasible satisfaction level t^* produces an optimal schedule by applying Algorithm 4.1 to the problem $n \mid m \mid I \mid L_{\max}$ with general due-dates $d'_{ij} = \mu_{ij}^{-1}(t^*)$ for i and j.

Next, checking $\lceil \log r \rceil$ t_i's for the feasibility by a binary search technique, we find the interval $[t_\rho, t_{\rho+1}]$ that includes t^* ($\lceil x \rceil$ means the smallest integer not less than x). Since the relative order of the modified due-dates $d'_{ij} = \mu_{ij}^{-1}(t)$ does not change when t varies in the interval, the structure of reduced network G does not change except that capacities s_i's change accoding to t. Furhter, note that minimum cuts in the network change, depending on s_i's. Accordingly, we focus on the interval that includes t^* and find subintervals (of the interval) such that minimum cuts in the network do not change. The subinterval can be found by using a similar idea to Megiddo [Meg1], i.e., by checking critical values t' (such that $[t_\rho, t']$ and $[t', t_{\rho+1}]$) of parameter t for feasibility when flow augmentation is computed by a maximal flow algorithm, e.g., Dinic's [Din]. That is, we check critical value t' such that minimum cuts in the reduced network with satisfaction level $t \in [t_\rho, t']$. Once the subinterval including t^* is

found, t^* is equal to the maximum t such that the capacity of minimum cuts is $\sum_{i=1}^{n} p_i$.

Now we are ready to describe the algorithm for solving $P6$ as follows.

Algorithm 4.2

Step 0. After setting $d'_{ij} := d_{ij}$ and constructing the reduced network $G = [\mathcal{N}; \mathcal{A}]$, find a maximum flow of G (let f be the resulting flow value). If $f = \sum_{i=1}^{n} p_i$, go to Step 4. Otherwise, proceed to Step 1.

Step 1. Compute $t_{(i,j),(k,l)}$ of (4.12) and sort them to obtain (4.13), i.e.,

$$t_0 = 0 < t_1 < \cdots < t_r < 1 = t_{r+1}.$$

After setting $IL := 0$ and $IU := r + 1$, go to Step 2.

Step 2. If $IU - IL = 1$, then go to Step 3. Otherwise, find a maximum flow (let f be the resulting flow value) of $G = [\mathcal{N}; \mathcal{A}]$ constructed on revised data, i.e., $t = \lfloor (IL + IU)/2 \rfloor$ and $d'_{ij} := \mu_{ij}^{-1}(t)$ for all i and j. If $f = \sum_{i=1}^{n} p_i$, then return to Step 2 after setting $IL := t$. Otherwise, return to Step 2 after setting $IU := t$.

Step 3. Find the maximum feasible satisfaction level t^* ($\in [t_{IL}, t_{IU}]$) by computing flow augmentation for G with $d'_{ij} := \mu_{ij}^{-1}(t_{IL})$. Go to Step 4.

Step 4. Based on the resulting flow with respect to t^*, we obtain an optimal schedule similarly to Step 5 in Algorithm 4.1.

Theorem 4.1 *Algorithm 4.2 for P6 finds an optimal schedule in* $O(\max(kGS, FS^2))$ *computational time, where GS is the complexity of Gonzalez and Sahni's*

algorithm applied to an $n \mid m \mid O \mid C_{\max}$ *preemptive scheduling problem, FS is the complexity of a maximum flow algorithm applied to the reduced network with* $O(k \max(m, n))$ *nodes and* $O(kmn)$ *arcs, and k is the maximum number of intervals* I_h *(note that k is at most mn).*

Proof. Since each membership function $\mu_{ij}(C_{ij}), i = 1, \cdots, n, j = 1, \cdots, m$ is the nonincreasing function of C_{ij}, it is clear that $\mu_{i,j(i)}(C_{i,j(i)}) \geq \alpha$ holds if and only if $C_{i,j(i)} \leq \mu_{i,j(i)}^{-1}(\alpha)$, where $j(i)$ is a machine index which completes the processing of job J_i finally. That is, $\mu_{i,j(i)}(C_{i,j(i)}) \geq \alpha$ holds if and only if there exists a feasible schedule with modified due-dates $d'_{ij} = \mu_{i,j(i)}^{-1}(\alpha)$.

From the observation with respect to reduced network G (described in Subsection 4.1.1), it is clear that there exists a feasible schedule that completes all jobs J_i on machine M_j by d'_{ij} if and only if there exists a maximal flow with flow value $\sum_{i=1}^{n} p_i$ (since the corresponding open shop problem [Gon] assures that the transformation of flow f_{ji}^h into the actual processing time assignment of job J_i to the machine M_j in the interval I_h is possible). To check the feasibility, it is necessary to compute maximal flow $\lceil \log r \rceil$ times, i.e., $O(FS \log(\max(m, n)))$ time, since $\lceil \log r \rceil$ is $O(\log(\max(m, n)))$. Moreover, the construction of a feasible schedule, if there exists any, for fixed d'_{ij} takes $O(kGS)$ time.

Now we focus on the complexity of finding t^* in interval $[t_\rho, t_{\rho+1}]$. Since t^* is the maximal value of t in the interval in which a feasible schedule is located, the capacity of its minimal cut is $\sum_{i=1}^{n} p_i$. We divide this interval into subintervals by a similar idea to [Meg1] where the minimal cuts of the network do not change. The subinterval including t^* can be found by the above method in $O(FS^2)$ time since we solve the maximum flow problem at each critical value. Once the subinterval is found, t^* can be found as the maximal value among

those t's (in the subinterval including t^*) such that the capacity of its minimal cut equals $\sum_{i=1}^{n} p_i$. This requires $O(\max(m,n))$ time.

In total, we can therefore find an optimal schedule in $O(\max(kGS, FS^2))$ time, since $O(FS)$ dominates $O(\max(m,n))$. □

4.2 Two Machine Open Shop Problem with Fuzzy Due-dates

4.2.1 Problem formulation

We investigate a generalized version of problem $n \mid 2 \mid O \mid \text{fuzzy}L_{\max}$. The word "generalized" means that the speed of each machine is controllable (see Subsection 1.3.3). In order to formulate this problem, we define some notations. Let s'_j be the speed of machine $M_j, j = 1,2$ and then $s_j \triangleq 1/s'_j$. Let a_i and b_i be the standard processing time on machines M_1 and M_2 at unit speed, respectively. Note that their actual processing time of J_i are $a_i s_1$ and $a_i s_2$, respectively. The membership function $\mu_i(C_i)$ denoting DM's degree of satisfaction with respect to the completion time C_i (of job J_i) is defined as

$$\mu_i(C_i) = \begin{cases} 1 & \text{if } C_i \le d_i, \\ 1 - \dfrac{C_i - d_i}{e_i} & \text{if } d_i < C_i \le d_i + e_i, \\ 0 & \text{if } C_i > d_i + e_i, \end{cases} \tag{4.15}$$

where d_i and e_i are nonnegative constants. Recall that each value d_i corresponds to the formal due-date in the original (ordinary) problem.

Now we formulate the problem as $P7$. The aim of $P7$ is to determine an optimal speed of each machine and an optimal schedule with respect to the objective function consisting of the sum of the minimum degree of satisfaction among all jobs and the cost of machine speeds. That is,

$$P7: \quad \text{Minimize} \quad -g_0(\min_i \mu_i(C_i)) + g_1 s'_1 + g_2 s'_2$$

$$\text{subject to } s'_1, s'_2 > 0, \tag{4.16}$$

where g_0, g_1 and g_2 are nonnegative constants. The value g_0 evaluates the gain that is attained by a unit increase of minimal satisfaction (actually this value may be fuzzy). The other constant g_1 (g_2) is the cost to speed up the processing of machine M_1 (M_2), which for example represents an extra cost of the corresponding electrical power and/or the cost for hiring some part-timers.

Now let

$$t \triangleq \min\{\mu_i(C_i) \mid i = 1,2,\cdots,n\} \tag{4.17}$$

be fixed, and then

$$C_i \le d_i + (1-t)e_i, \text{ for all } i \tag{4.18}$$

must hold (similarly to (3.2) through (3.4) in Chapter 3). The inequalities (4.18) mean that each $d_i + (1-t)e_i$ play the role of the due-date in the ordinary sense. Therefore, if there exist a feasible schedule satisfying these due-dates, we can assume without loss of generality that it processes jobs in nondecreasing order of due-dates [Law1]. We express such order by π (i.e., $\pi(j)$ is the index of the $j-$th job in the schedule), and the feasibility of this schedule π can be checked as follows [Law1].

Suppose that $J_{\pi(1)}, J_{\pi(2)}, \cdots, J_{\pi(i-1)}$ have been successfully scheduled. Now job $J_{\pi(i)}$ has to be scheduled. First we define that

$$A_i^\pi = \sum_{k=1}^{i} a_{\pi(k)}, \; B_i^\pi = \sum_{k=1}^{i} b_{\pi(k)}, \quad i = 1,2,\cdots,n. \tag{4.19}$$

Let $x_{\pi(i)}$ ($y_{\pi(i)}$) denote the total amount of time prior to $d_{\pi(i)} + (1-t)e_{\pi(i)}$ in which machine M_1 (M_2) is idle while M_2 (M_1) is busy, and let $z_{\pi(i)}$ denote the total amount of time prior to $d_{\pi(i)} + (1-t)e_{\pi(i)}$ in which M_1 and M_2 are simultaneously idle. Note that $x_{\pi(i)}, y_{\pi(i)}$ and $z_{\pi(i)}$ are not independent,

inasmuch as

$$x_{\pi(i)} + z_{\pi(i)} = d_{\pi(i)} + (1-t)e_{\pi(i)} - A^{\pi}_{i-1}s_1, \tag{4.20}$$

$$y_{\pi(i)} + z_{\pi(i)} = d_{\pi(i)} + (1-t)e_{\pi(i)} - B^{\pi}_{i-1}s_2. \tag{4.21}$$

The minimum amount of operation $O_{\pi(i),1}$ ($O_{\pi(i),2}$) that must be processed on M_1 (M_2) while both machines are available is

$$\max\left\{0, s_1a_{\pi(i)} - x_{\pi(i)}\right\}\ \left(\max\left\{0, s_2b_{\pi(i)} - y_{\pi(i)}\right\}\right)$$

(for details; see [Law1]). It follows that $J_{\pi(i)}$ can be successfully scheduled if and only if

$$\max\left\{0, s_1a_{\pi(i)} - x_{\pi(i)}\right\} + \max\left\{0, s_2b_{\pi(i)} - y_{\pi(i)}\right\} \le z_{\pi(i)}. \tag{4.22}$$

This inequality (4.22) is equivalent to the following four inequalities:

$$\begin{aligned} &0 \le z_{\pi(i)}, \\ &s_1a_{\pi(i)} - x_{\pi(i)} \le z_{\pi(i)}, \\ &s_2b_{\pi(i)} - y_{\pi(i)} \le z_{\pi(i)}, \\ &s_1a_{\pi(i)} - x_{\pi(i)} + s_2b_{\pi(i)} - y_{\pi(i)} \le z_{\pi(i)}. \end{aligned} \tag{4.23}$$

Discarding the first of these inequalities as vacuous and applying (4.20) and (4.21) to the remaining ones (because the resulting (4.27) assures nonnegativity of $z_{\pi(i)}$), we see that $J_{\pi(i)}$ can be successfully scheduled if and only if each of the following feasibility conditions holds [Law1].

$$s_1A^{\pi}_i \le d_{\pi(i)} + (1-t)e_{\pi(i)}, \tag{4.24}$$

$$s_2B^{\pi}_i \le d_{\pi(i)} + (1-t)e_{\pi(i)}, \tag{4.25}$$

$$s_1A^{\pi}_i + s_2B^{\pi}_i \le 2(1-t)e_{\pi(i)} - z_{\pi(i)}, \tag{4.26}$$

where $z_{\pi(1)}, z_{\pi(2)}, \cdots, z_{\pi(n)}$ are defined recursively by

$$\begin{aligned} z_{\pi(1)} &= d_{\pi(1)} + (1-t)e_{\pi(1)}, \\ z_{\pi(i)} &= d_{\pi(i)} - d_{\pi(i-1)} + (1-t)\left\{e_{\pi(i)} - e_{\pi(i-1)}\right\} \\ &\quad + \max\left\{0, z_{\pi(i-1)} - s_1a_{\pi(i-1)} - s_2b_{\pi(i-1)}\right\}, \\ &\quad i = 2, 3, \cdots, n. \end{aligned} \tag{4.27}$$

Solving recursive relation (4.27), we obtain

$$\begin{aligned} z_{\pi(i)} = \max_{1\le k\le i}\Bigg\{ &d_{\pi(i)} - d_{\pi(i-k)} \\ &+ (1-t)\left(e_{\pi(i)} - e_{\pi(i-k)}\right) \\ &- s_1\sum_{l=1}^{k} a_{\pi(i-l)} - s_2\sum_{l=1}^{k} b_{\pi(i-l)}\Bigg\}, \\ i = 1, 2, \cdots, n, \end{aligned} \tag{4.28}$$

where we define $d_{\pi(0)} = e_{\pi(0)} = 0$. Substituting the above $z_{\pi(i)}$ into (4.26), we rewrite the feasibility conditions of (4.24) through (4.26) as follows:

$$\begin{aligned} s_1A^{\pi}_i &\le d_{\pi(i)} + (1-t)e_{\pi(i)}, \\ s_2B^{\pi}_i &\le d_{\pi(i)} + (1-t)e_{\pi(i)}, \end{aligned}$$

$$\begin{aligned} s_1A^{\pi}_i + s_2B^{\pi}_i \le \min_{1\le k\le i}\Bigg\{ &d_{\pi(i)} + d_{\pi(i-k)} \\ &+ (1-t)(e_{\pi(i)} + e_{\pi(i-k)}) \\ &+ s_1\sum_{l=1}^{k-1} a_{\pi(i-l)} + s_2\sum_{l=1}^{k-1} b_{\pi(i-l)}\Bigg\}, \\ i = 1, 2, \cdots, n, \end{aligned} \tag{4.29}$$

where we define $\sum_{l=1}^{0} a_{\pi(i-l)} = \sum_{l=1}^{0} b_{\pi(i-l)} = 0$. From these inequalities, $P7$ is equivalent to the following problem $P7'$:

$$P7': \quad \text{Minimize } g_0\bar{t} + g_1/s_1 + g_2/s_2$$

subject to

$$s_1 A_i^{\pi} \le d_{\pi(i)} + \bar{t} e_{\pi(i)},$$

$$s_2 B_i^{\pi} \le d_{\pi(i)} + \bar{t} e_{\pi(i)},$$

$$s_1 A_i^{\pi} + s_2 B_i^{\pi} \le \min_{1 \le k \le i} \left\{ d_{\pi(i)} + d_{\pi(i-k)} + \bar{t}(e_{\pi(i)} + e_{\pi(i-k)}) + s_1 \sum_{l=1}^{k-1} a_{\pi(i-l)} + s_2 \sum_{l=1}^{k-1} b_{\pi(i-l)} \right\},$$

$$s_1, s_2 > 0 \text{ and } 0 \le \bar{t} \le 1, \tag{4.30}$$

where $\bar{t} = 1 - t$.

The order of $d_i + e_i \bar{t}$ for all i changes according to the value of $\bar{t}$. We therefore define

$$\bar{t}_{ij} = \frac{d_i - d_j}{e_j - e_i}, \tag{4.31}$$
$$i < j,\ i = 1, 2, \cdots, n-1, j = 2, 3, \cdots, n,$$

and sort different $\bar{t}_{ij}$'s such that $0 \le \bar{t}_{ij} \le 1$ as follows:

$$\bar{t}_0 \triangleq 0 < \bar{t}_1 < \cdots < 1 \triangleq \bar{t}_q, \tag{4.32}$$

where $\bar{t}_l$ denotes the l-th smallest value among $\bar{t}_{ij}$'s. By dividing interval $[0,1]$ into subintervals $T_l = [\bar{t}_{l-1}, \bar{t}_l], l = 1, 2, \cdots, q$, we introduce the following subproblem $P7_l, l = 1, 2, \cdots, q$, since the ordering of $d_i + e_i \bar{t}$ for all i does not change over the interval T_l.

$P7_l$: Minimize $g_0 \bar{t} + g_1/s_1 + g_2/s_2$

subject to

$$s_1 A_i^{\pi_l} \le d_{\pi_l(i)} + \bar{t} e_{\pi_l(i)}, \tag{4.33}$$

$$s_2 B_i^{\pi_l} \le d_{\pi_l(i)} + \bar{t} e_{\pi_l(i)}, \tag{4.34}$$

$$s_1 \left\{ A_i^{\pi_l} - \sum_{m=1}^{k-1} a_{\pi_l(i-m)} \right\} + s_2 \left\{ B_i^{\pi_l} - \sum_{m=1}^{k-1} b_{\pi_l(i-m)} \right\} \le \left\{ e_{\pi_l(i)} + e_{\pi_l(i-k)} \right\} \bar{t} + d_{\pi_l(i)} + d_{\pi_l(i-k)}, \tag{4.35}$$

$$k = 1, 2, \cdots, i,\ i = 1, 2, \cdots, n,$$

$$s_1, s_2 > 0,\ \bar{t} \in T_l,$$

where π_l denotes the ordering of indices i according to $d_i + e_i \bar{t}$ for $\bar{t} \in T_l$.

Thus, by using the solution procedure (discussed in the succeeding subsection) for solving all the subproblems, we can obtain an optimal solution of $P7$ as the best solution among the optimal solutions of $P7_1, P7_2, \cdots, P7_q$.

4.2.2 Solution procedure for $P7_l$

For simplicity of presentaion, we assume that $\pi_l(i) = i$ by changing job indicies if necessary, i.e.,

$$d_i + e_i \bar{t} \le d_{i+1} + e_{i+1} \bar{t}, \quad i = 1, 2, \cdots, n$$

for $\bar{t} \in T_l$. Since at least one inequality among (4.33) through (4.35) holds as equality, we may set

$$\bar{t} = \max \left\{ U_j s_1 + V_j s_2 + W_j \mid 1 \le j \le \frac{1}{2}\left(n^2 + 5n\right) \right\}, \tag{4.36}$$

where

$$U_j = A_j^{\pi_l}/e_j,\ V_j = 0, W_j = -d_j/e_j \quad (1 \le j \le n),$$

$$U_j = 0,\ V_j = B_{j-n}^{\pi_l}/e_{j-n}, W_j = -d_{j-n}/e_{j-n} \quad (n+1 \le j \le 2n),$$

and

$$U_j = \left(A_i^{\pi_l} - \sum_{m=1}^{k-1} a_{i-m}\right) \Big/ (e_i + e_{i-k}),$$
$$V_j = \left(B_i^{\pi_l} - \sum_{m=1}^{k-1} b_{i-m}\right) \Big/ (e_i + e_{i-k}),$$
$$W_j = -(d_i + d_{i-k}) / (e_i + e_{i-k})$$
$$(j = 2n + i(i-1)/2 + k,\ k = 1, 2, \cdots, i,\ i = 1, 2, \cdots, n),$$

In order to solve $P7_l$, we further introduce the following auxiliary problems $Q_j, j = 1, 2, \cdots, (n^2 + 5n)/2$, where we assume that there are no redundant inequalities among (4.33) through (4.35) without any loss of generality.

$$\begin{aligned} Q_j: \quad & \text{Minimize } g_0 (U_j s_1 + V_j s_2 + W_j) + g_1/s_1 + g_2/s_2 \\ & \text{subject to} \\ & U_{jk} s_1 + V_{jk} s_2 + W_{jk} \le 0,\ k = 1, 2, \cdots, \frac{1}{2}\left(n^2 + 5n\right), \\ & s_1, s_2 > 0, \end{aligned} \tag{4.37}$$

where

$$U_{jk} = -U_j + U_k,\ V_{jk} = -V_j + V_k,\ W_{jk} = -W_j + W_k$$
for $k \neq j$,
$$U_{jj} = U_j,\ V_{jj} = V_j,\ W_{jj} = W_j - \bar{t}_l,$$
$$U_{j0} = -U_j,\ V_{j0} = -V_j,\ W_{j0} = -W_j - \bar{t}_{l-1}.$$

By solving all these auxiliary problems, we can determine optimal $\bar{t}$ and (s_1, s_2) of $P7_l$ as the minimum value among optimal values of $Q_1, \cdots, Q_{(n^2+5n)/2}$.

Based on the above observation, we describe how to solve each Q_j by a similar idea to [Meg2] as follows.

Theorem 4.2 *Each Q_j is a convex programming problem. The optimal value $z(s_1)$ of Q_j for a fixed s_1 is a convex function of s_1.*

Proof. For $(s_1^\alpha, s_2^\alpha), (s_1^\beta, s_2^\beta)$ satisfying both the inequalities in (4.37), let

$$s_1^\lambda = \lambda s_1^\alpha + \overline{\lambda} s_1^\beta,\ s_2^\lambda = \lambda s_2^\alpha + \overline{\lambda} s_2^\beta, \tag{4.38}$$

for $1 \le \lambda \le 1$ and $\overline{\lambda} = 1 - \lambda$. Then

$$\begin{aligned} U_{jk} s_1^\lambda + V_{jk} s_2^\lambda + W_{jk} &= \lambda \left(U_{jk} s_1^\alpha + V_{jk} s_2^\alpha + W_{jk}\right) \\ &\quad + \overline{\lambda} \left(U_{jk} s_1^\beta + V_{jk} s_2^\beta + W_{jk}\right) \le 0, \end{aligned} \tag{4.39}$$

that is, $(s_1^\lambda, s_2^\lambda)$ also satisfies (4.37). This shows that the feasible region of Q_j is a convex set. Since the convexity of the objective function is clear, Q_j is a convex programming problem.

Now let s_2^α (s_2^β) denote an optimal value of s_2 when $s_1 = s_1^\alpha$ (s_1^β). Furthermore, let $(s_1^\lambda, s_2^\lambda)$ be defined as in (4.38). Then

$$\begin{aligned} z(s_1) &\le g_0 \left(U_j s_1^\lambda + V_j s_2^\lambda + W_j\right) + g_1/s_1^\lambda + g_2/s_2^\lambda \\ &= \lambda g_0 \left(U_j s_1^\alpha + V_j s_2^\alpha + W_j\right) \\ &\quad + \overline{\lambda} \left(U_j s_1^\beta + V_j s_2^\beta + W_j\right) \\ &\quad + g_1 \Big/ \left(\lambda s_1^\alpha + \overline{\lambda} s_1^\beta\right) + g_2 \Big/ \left(\lambda s_2^\alpha + \overline{\lambda} s_2^\beta\right) \\ &\le \lambda \left\{g_0 \left(U_j s_1^\alpha + V_j s_2^\alpha + W_j\right) + g_1/s_1^\alpha + g_2/s_2^\alpha\right\} \\ &\quad \overline{\lambda} \left\{g_0 \left(U_j s_1^\beta + V_j s_2^\beta + W_j\right) + g_1/s_1^\beta + g_2/s_2^\beta\right\} \\ &= \lambda z(s_1^\alpha) + \overline{\lambda} z(s_1^\beta) \end{aligned} \tag{4.40}$$

since s_2^λ is not necessarily an optimal value for s_1^λ, and the function $1/x$ is convex. This shows the convexity of $z(s_1)$. □

When we deal with the index k of V_{jk} as

$$I_1 = \{k \mid V_{jk} > 0\},\ I_2 = \{k \mid V_{jk} < 0\},$$

and solve (4.37) with respect to s_2, we can rewrite the constraints in (4.37) as follows:

$$\max_{k\in I_2}\{f_k s_1 + h_k\} \le s_2 \le \min_{k\in I_1}\{f_k s_1 + h_k\}, \tag{4.41}$$

$$\underline{s}_1 \le s_1 \le \overline{s}_1, \tag{4.42}$$

$$\underline{s}_2 \le s_2 \le \overline{s}_2, \tag{4.43}$$

where f_k is the coefficient of s_1 and h_k is constant term, i.e., $f_k = -U_{jk}/V_{jk}$ and $h_k = -W_{jk}/V_{jk}$. Moreover $\underline{s}_1, \overline{s}_1, \underline{s}_2, \overline{s}_2$ are obtained by the inequalities of (4.37) with $U_{jk} = 0$ or $V_{jk} = 0$ respectively. Consider the two inequalities

$$s_2 \le f_i s_1 + h_i,\ s_2 \le f_k s_1 + h_k,$$

for $i, k \in I_1$. If $f_i \ge f_k$ and $h_i \ge h_k$ hold, the constraint $s_2 \le f_i s_1 + h_i$ is redundant. Otherwise, let $L_{ik} = (h_k - h_i)/(f_i - f_k)$. If $L_{ik} \ge \overline{s}_1$ or $L_{ik} \le \underline{s}_1$, one of these inequalities is redundant again. If $\underline{s}_1 < L_{ik} < \overline{s}_1$, L_{ik} divides (4.42) into two intervals $[\underline{s}_1, L_{ik}]$ and $[L_{ik}, \overline{s}_1]$. Once we know that an optimal solution to Q_j (if any) lies in an interval determined by L_{ik}, then we may discard remaining one of the two inequalities. A similar observation is true for every pair of inequalities

$$s_2 \ge f_i s_1 + h_i,\ s_2 \ge f_k s_1 + h_k,$$

for $i, k \in I_2$.

Now we start the procedure for Q_j by combining inequalities in arbitary disjoint pairs such that the two members of each pair belong to the same set $I_m, m = 1, 2$. For each pair of two inequalities, either we can drop one of the participating inequalities based on the above observation or we have a dividing point L_{ik}. Consider the set of dividing points $\{s_1^1, s_1^2, \cdots, s_1^{k_1}\}$ that are

generated in this way and find the median s_M among them. If the subdifferential of $z(s_1)$ at $s_1 = s_M$ includes 0, then an optimal solution $(\tilde{s}_1, \tilde{s}_2)$ of Q_j is obtained as s_M and the value of s_2 that realizes $z(s_M)$, respectively. If all subgradients of $z(s_1)$ at $s_1 = s_M$ are negative, we update $\underline{s}_1 = s_M$ and for each of dividing points less than s_M, at least one inequality can be interpreted as redundant and discarded as inactive. Consequently, at least one fourth among all inequalities can be discarded. If all subgradients of $z(s_1)$ at $s_1 = s_M$ are positive, we update $\overline{s}_1 = s_M$ and again at least one fourth of all inequalities can be discarded similarly. By repeating the above operations to the remaining inequalities, the set of inequalities (4.41) through (4.43) is reduced to the following three inequalities.

$$f_{\underline{k}} s_1 + h_{\underline{k}} \le s_2 \le f_{\overline{k}} s_1 + h_{\overline{k}}, \tag{4.44}$$

$$\underline{s}_1 \le s_1 \le \overline{s}_1, \tag{4.45}$$

$$\underline{s}_2 \le s_2 \le \overline{s}_2, \tag{4.46}$$

If

$$(s_1, s_2) = \left(\sqrt{\frac{g_1}{g_0 U_j}}, \sqrt{\frac{g_2}{g_0 V_j}}\right) \tag{4.47}$$

satisfies (4.44) through (4.46), then this (s_1, s_2) is an optimal solution of Q_j. Otherwise, one of the inequalities (4.44) - (4.46) holds as equality. In that case, Q_j reduces to an optimization problem of one variable and hence an optimal solution of Q_j can be obtained as the solution of a quadratic equation.

Now we are ready to describe the algorithm for solving Q_j. In the algorithm, DP is prepared to store k_1 dividing points L_{ik}'s for I_1. The complexity of the algorithm is discussed in the last theorem of this chapter.

Algorithm 4.3

Step 0. Formulate auxiliary problems Q_j of (4.37) and drop redundant inequalities. Let the dividing points L_{ik} for I_1 as a result be $\{s_1^1, \cdots, s_1^{k_1}\}$ and store these values into DP. Proceed to Step 1.

Step 1. Find the median s_M among the elements in DP and proceed to Step 2.

Step 2. If $\partial z(s_M) \ni 0$ (i.e., the subdifferential of $z(s_1)$ at $s_1 = s_M$ includes 0), an optimal solution

$$\tilde{s}_1 := s_M,$$

$$\tilde{s}_2 := \text{ the value of } s_2 \text{ that realizes } z(s_M),$$

is obtained and terminate. Otherwise, it is divided into the following two cases, i.e.,

1. if all $\partial z(s_M) < 0$, update $\underline{s}_1 := s_M$ and the corresponding redundant inequalities are discarded,
2. if all $\partial z(s_M) > 0$, update $\overline{s}_1 := s_M$ and the corresponding redundant inequalities are discarded,

and if the remaining inequalities are reduced to three equations (4.44)-(4.46), then proceed to Step 3. Otherwise, return to Step 1 after updating the values of DP, namely calculating L_{ik} for I_1 according to the remaining inequalities.

Step 3. If equation (4.47) satisfies all the inequalities (4.44) through (4.46), (4.47) is an optimal solution of Q_j. Otherwise, Q_j reduces to an optimization problem of one variable (since one of (4.44)-(4.46) holds as equality).

4.2.3 Solution procedure for $P7'$

By fully utilizing the results in the previous Subsection 4.2.2, each subproblem $P7_l$ can be solved by solving all of its auxiliary problems. Then an optimal solution of $P7'$ is constructed from the best solution among the optimal solutions of these subproblems as follows.

Let the best solution be obtained from an optimal solution $\overline{t}^*_{\pi_{l^*}}$ and $(s_1^{\pi_{l^*}}, s_2^{\pi_{l^*}})$ of $P7_{l^*}$. Then an optimal scheduling order of our problem is π_{l^*} and optimal machine speed is

$$(s_1^*, s_2^*) = \left(s_1^{\pi_{l^*}}, s_2^{\pi_{l^*}}\right).$$

Moreover, the corresponding minimal satisfaction is $t^* = 1 - \overline{t}^*_{\pi_{l^*}}$. The actual optimal schedule can be obtained by applying the result of [Law1] to the ordinary two machine maximum lateness problem with job processing times

$$(a_i s_1^*, b_i s_2^*), \quad i = 1, 2, \cdots, n$$

and due-dates

$$d_i = e_i(1 - t^*), \quad i = 1, 2, \cdots, n.$$

Theorem 4.3 *If a quadratic equation can be solved in $O(1)$ computational time, our procedure solves $P7'$ in $O(n^6)$ time.*

Proof. First we show the complexity of Algorithm 4.3 for solving Q_j. In order to reduce (4.41)-(4.43) into (4.44)-(4.46), first $O(n^2)\, L_{ik}$, second $O(\frac{3}{4}n^2)\, L_{ik}$, $\cdots$ and in total $O(n^2)$ linear equations must be solved. It is known that the median of $O(n^2)$ elements is found on $O(n^2)$ computational time [Aho]. The subdifferential of $z(s_1)$ at $s_1 = s_M$ can be determined by its right derivative and left derivative, which are calculated by some perturbation method. Unconstrained minimization of the objective function of Q_j for a fixed s_1 is

always realized by $s_2 = \sqrt{g_2/(g_0 V_j)}$ and the corresponding feasible region of s_2 is a closed interval, which is constructed in $O(m)$ time when there remain $O(m)$ inequalities. This implies that each subdifferential can be calculated in $O(m)$ time. Thus the total computational time for the subdifferentials is again $O(n^2)$. After the calculation of each differential, either an optimal solution of Q_j is obtained, or one fourth of the constraints is discarded. Consequently the total computational time for Q_j is $O(n^2)$ if a quadratic equation can be solved in $O(1)$ computational time.

The number of auxiliary problems is $O(n^2)$ and that of subproblems is $O(n^2)$. Therefore, in total, $P7'$ can be solved in $O(n^6)$ computational time by our procedure. □

Part II

FUZZY NETWORK PROBLEMS

Chapter 5

FUZZY SHARING PROBLEM

5.1 Intorduction

As mentioned in Subsection 1.4.3 (Paragraph A), the purpose of the sharing problem is to find an optimal flow in a capacitated network, which realizes an equitable distribution of some resource. An equitable distribution is defined in terms of the weights given to sink nodes of the network. For example, when we interpret the sink nodes as some factories (or plants) that recieve a share of resource, the weights of sink nodes represent the relative scales of the corresponding factories. Then the objective is to maximize the smallest value among trade off values defined as the quotient of flow values and their weights for the sink nodes (see (1.21) in Subsection 1.4.3).

In some situation, however, a factory may be satisfied with a share that is less than the exact optimal value since the requirements of factories are usually stated in a fuzzy manner and accordingly a single value of each weight may unfit for the corresponding requirement. That is, it is difficult to determine just (equitable) weights rigidly, and they may be contradictory with each other. In order to face up to such a difficulty, we consider the problem as a decision making problem in fuzzy environment (i.e., the problem with fuzzy weights).

For instance, three factories A, B and C may require that "A needs a large share if possible", "B can't be satisfied with values less than a certain value" and "a certain value is sufficient for C", respectively. As a result, these weights may not be rigidly determined. In this case, it is natural to express them by membership functions, leading to the sharing problem with fuzzy weights.

In this chapter, we consider two fuzzy versions of the sharing problem: the fuzzy sharing problem $P8$ and a generalized version of the fuzzy sharing problem $P9$. The latter $P9$ is a generalization of $P8$ in the sense that it incorporates the "realistic share constraint", such that every share must be some "block-unit" (e.g., if the unit is "a dozen", every share is 12, 24, 36, or $\cdots$), into the former $P8$.

5.2 Fuzzy sharing problem

5.2.1 Formulation and solution procedure

In order to formulate $P8$, we introduce membership functions $\mu_{t_j}(f(t_j))$ (which are simplified notations of $\mu_{\boldsymbol{w}(t_j)}(f(t_j))$), $j = 1, 2, \cdots, l$ characterizing fuzzy weights $\boldsymbol{w}(t_j)$:

$$\mu_{t_j}(f(t_j)) = \begin{cases} 0 & \text{if } f(t_j) \leq a_{t_j}, \\ \dfrac{f(t_j) - a_{t_j}}{b_{t_j} - a_{t_j}} & \text{if } a_{t_j} < f(t_j) < b_{t_j}, \\ 1 & \text{if } f(t_j) \geq b_{t_j}. \end{cases} \tag{5.1}$$

Here a_{t_j} and b_{t_j} are positive real numbers that embody the requirements of sink nodes (see Fig.5.1). Recall that t_j's denote the l sink nodes ($T = \{t_1, \cdots, t_l\}$) and $f(t_j)$ total amount of flows sent into a sink t_j. We formulate the fuzzy sharing problem by replacing the trade-off function of the original problem (see Paragraph A in Subsection 1.4.3) with the membership functions, that is,

$$P8: \quad \text{Maximize} \quad \min_{t_j \in T} \mu_{t_j}(f(t_j)) \tag{5.2}$$

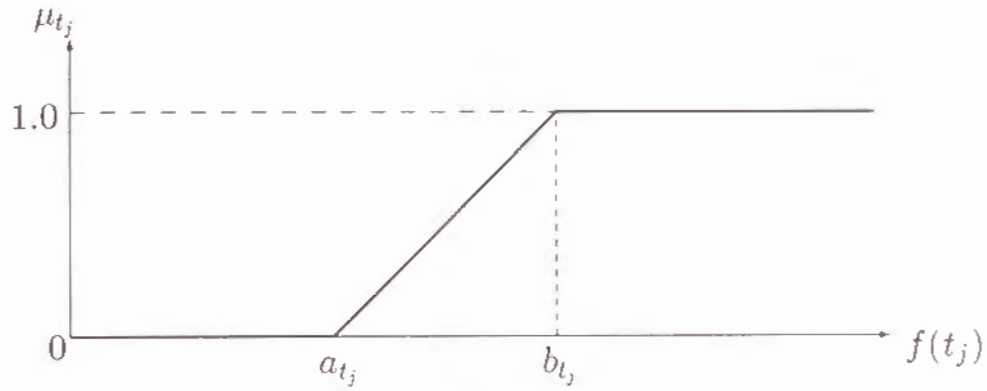

Fig. 5.1 Membership function μ_{t_j}

subject to

$$\sum_{y \in A(x)} f(x,y) - \sum_{y \in B(x)} f(y,x) = \begin{cases} v^* & \text{if } x = s \\ 0 & \text{if } x \neq s, t \\ -v^* & \text{if } x = t, \end{cases} \tag{5.3}$$

$$0 \leq f(x,y) \leq c(x,y), \ x \in N' - \{t\}, y \in N' - \{s\}. \tag{5.4}$$

Here recall that v^* denotes the total amount of resources and that N' is a set of all nodes including super source and super sink, i.e., $N' = \mathcal{N} + \{s\} + \{t\}$ (see Subsection 1.4.2 for the notations $A(x)$ and $B(x)$). The objective of (5.2) is to maximize the smallest value among all membership values $\mu_{t_j}(f(t_j))$ (we call it *max-min sharing*). Incidentally, if we minimize the greatest value of all membership values, it is called *min-max sharing*. The objective should be two criteria that is not only "max-min" but also "min-max", in case we strictly consider the most equitable distribution. However, we shall give priority to the criterion of "max-min" and additionally consider the other "min-max" if possible, since it may become rather difficult if we consider both the criteria simultaneously.

We now outline Algorithm 5.1 for solving $P8$ (its validity and complexity will be discussed in Subsection 5.2.2). When we solve the problem, the most equitable distribution is attained if every sink node t_j $(j = 1, \cdots, l)$ derives a same degree of satisfaction, i.e, $\mu_{t_j}(f(t_j)) = \alpha$, where α is a constant and $\sum_{j=1}^{l} f(t_j) = v^*$ holds. Based on the constants a_{t_j} and b_{t_j}, we can calculate such "ideal shares" (i.e., $\mu_1'(b_{t_j} - a_{t_j}) + a_{t_j}$ in Step 1 of the algorithm). After setting each value of the ideal shares as capacities from sink t_j to super sink t, we find a maximum flow from super source s into super sink t together with the corresponding minimum cut $(X_h, \overline{X}_h)$ such that $s \in X_h$ and $t \in \overline{X}_h$ (h is an integer that indicates times of iteration). If the total flow value is equal to the total amount of resource v^*, then the flow is optimal, namely, the most equitable sharing is realized (see Subsection 5.2.2) and the algorithm consequently terminates. Otherwise, since the total value of the minimum cut (capacity) is less than v^*, we update each value of capacities $c(t_j, t)$ according to the information whether sink nodes t_j belong to X_h or $\overline{X}_h$. In case $t_j \in X_h$ $(t_j \in \overline{X}_h)$, the capacities $c(t_j, t)$ is increased (decreased). That is, the capacities need to be updated in order for the first constraint in (5.3) (namely, the total value of max-flow must he equal to v^*) to hold. At this time, we utilize the resultant minimum cut to take account of "max-min sharing" and "min-max sharing" for the sink nodes in $\overline{X}_h$ and X_h, respectively (in Step 4 at the first iteration and in Step 5 otherwise). After the updates of all $c(t_j, t)$, we find a max-flow for the network (in Step 2 at the next iteration). The algorithm is repeated until the total value of max-flow is equal to v^*.

Algorithm 5.1

Step 1. Set $h := 1$, $c(s, s_i) := \infty$, $i = 1, \cdots, k$, and define the following:

$$\mu_1' := \left(v^* - \sum_{t_j \in T} a_{t_j}\right) \Big/ \sum_{t_j \in T} \left(b_{t_j} - a_{t_j}\right).$$

If $\mu_1' \leq 0$ then stop (the optimal value is 0); otherwise set $c(t_j, t) := \mu_1'(b_{t_j} - a_{t_j}) + a_{t_j}$, $j = 1, \cdots, l$.

Step 2. Find a maximum flow f_h from s to t together with its value v_h and the corresponding minimum cut $(X_h, \overline{X}_h)$, $s \in X_h, t \in \overline{X}_h$.

Step 3. If $v_h < v^*$ then go to Step 4. Otherwise stop, i.e., the current flow is optimal.

Step 4. If $h \geq 2$ then go to Step 5 (namely, Step 4 is executed only at the first iteration $h = 1$). Moreover if $| X_1 \cap T | = 0$ then stop; this problem is infeasible. Otherwise after calculating

$$\mu'_{\overline{X}_1} := \left(F_{\overline{X}_1} - \sum_{t_j \in \overline{X}_1 \cap T} a_{t_j}\right) \Big/ \sum_{t_j \in \overline{X}_1 \cap T} \left(b_{t_j} - a_{t_j}\right),$$

$$\mu'_{X_1} := \left(v^* - F_{\overline{X}_1} - \sum_{t_j \in X_1 \cap T} a_{t_j}\right) \Big/ \sum_{t_j \in X_1 \cap T} \left(b_{t_j} - a_{t_j}\right),$$

where $F_{\overline{X}_1} = \sum_{t_j \in \overline{X}_1 \cap T} f(t_j)$, each capacity from t_j to t is updated as follows:

$$c(t_j, t) := \mu'_{\overline{X}_1}(b_{t_j} - a_{t_j}) + a_{t_j}, \quad \text{for } t_j \in \overline{X}_1 \cap T,$$
$$c(t_j, t) := \mu'_{X_1}(b_{t_j} - a_{t_j}) + a_{t_j}, \quad \text{for } t_j \in X_1 \cap T.$$

Finally set $h := 2$ and return to Step 2.

Step 5. For each sink node included in $\overline{X}_1$, compute

$$\mu'_{\overline{X}_1 \cap \overline{X}_h} := \left(F_{\overline{X}_1 \cap \overline{X}_h} - \sum_{t_j \in \overline{X}_1 \cap \overline{X}_h \cap T} a_{t_j}\right) \Big/ \sum_{t_j \in \overline{X}_1 \cap \overline{X}_h \cap T} \left(b_{t_j} - a_{t_j}\right),$$

$$\mu'_{\overline{X}_1\cap X_h} := \left(F_{\overline{X}_1} - F_{\overline{X}_1\cap\overline{X}_h} - \sum_{t_j\in\overline{X}_1\cap X_h\cap T} a_{t_j}\right) \Big/ \sum_{t_j\in\overline{X}_1\cap X_h\cap T}\left(b_{t_j}-a_{t_j}\right),$$

where $F_{\overline{X}_1\cap\overline{X}_h} = \sum_{t_j\in\overline{X}_1\cap\overline{X}_h\cap T} f(t_j)$ and $F_{\overline{X}_1\cap X_h} = \sum_{t_j\in\overline{X}_1\cap X_h\cap T} f(t_j)$. Each capacity from t_j to t is updated as follows:

$$c(t_j,t) := \mu'_{\overline{X}_1\cap\overline{X}_h}(b_{t_j}-a_{t_j})+a_{t_j}, \quad \text{for } t_j\in\overline{X}_1\cap\overline{X}_h\cap T,$$
$$c(t_j,t) := \mu'_{\overline{X}_1\cap X_h}(b_{t_j}-a_{t_j})+a_{t_j}, \quad \text{for } t_j\in\overline{X}_1\cap X_h\cap T.$$

For each sink node included in X_1, compute

$$\mu'_{X_1\cap\overline{X}_h} := \left(F_{X_1\cap\overline{X}_h} - \sum_{t_j\in X_1\cap\overline{X}_h\cap T} a_{t_j}\right) \Big/ \sum_{t_j\in X_1\cap\overline{X}_h\cap T}\left(b_{t_j}-a_{t_j}\right),$$

$$\mu'_{X_1\cap X_h} := \left(v^* - F_{\overline{X}_1} - F_{X_1\cap\overline{X}_h} - \sum_{t_j\in X_1\cap X_h\cap T} a_{t_j}\right) \Big/ \sum_{t_j\in X_1\cap X_h\cap T}\left(b_{t_j}-a_{t_j}\right),$$

where $F_{X_1\cap\overline{X}_h} = \sum_{t_j\in X_1\cap\overline{X}_h\cap T} f(t_j)$, and update the capacities of the corresponding sink nodes as follows:

$$c(t_j,t) := \mu'_{X_1\cap\overline{X}_h}(b_{t_j}-a_{t_j})+a_{t_j}, \quad \text{for } t_j\in X_1\cap\overline{X}_h\cap T,$$
$$c(t_j,t) := \mu'_{X_1\cap X_h}(b_{t_j}-a_{t_j})+a_{t_j}, \quad \text{for } t_j\in X_1\cap X_h\cap T.$$

Finally, set $h := h+1$ and return to Step 2.

5.2.2 Validity and complexity of Algorithm 5.1

Here we show the validity and complexity of Algorithm 5.1. First of all, it is clear that the setting of the ideal shares $\mu'_1(b_{t_j}-a_{t_j})+a_{t_j}$ in Step 1 of the

algorithm is valid, since all the corresponding membership functions indicate the same grade μ'_1 and the sum of the ideal shares is equal to the total amount of resource v^*. That is, we have

$$\begin{aligned}\mu_{t_j}(\mu'_1(b_{t_j}-a_{t_j})+a_{t_j}) &= \frac{\mu'_1(b_{t_j}-a_{t_j})+a_{t_j}-a_{t_j}}{b_{t_j}-a_{t_j}}\\ &= \mu'_1, \end{aligned} \tag{5.5}$$

from (5.1) and

$$\begin{aligned}\sum_{j=1}^{l}\{\mu'_1(b_{t_j}-a_{t_j})+a_{t_j}\} &= \mu'_1\sum_{t_j\in T}(b_{t_j}-a_{t_j})+\sum_{t_j\in T}a_{t_j}\\ &= \frac{v^*-\sum_{t_j\in T}a_{t_j}}{\sum_{t_j\in T}(b_{t_j}-a_{t_j})}\sum_{t_j\in T}(b_{t_j}-a_{t_j})+\sum_{t_j\in T}a_{t_j}\\ &= v^*. \end{aligned} \tag{5.6}$$

Note that setting these ideal shares as $c(t_j,t)$ (in Step 1) means that we try to find a maximum flow (from s to t) such that $f(t_j)=c(t_j,t)$ at the first iteration (and hence $v_1 \not> v^*$ holds). If there exists such a flow, i.e., if $v_1 = v^*$, the current flow is optimal. That is, in this case, if the total flow value into any sink node is increased (decreased) then that into some sink node must be decreased (increased) because of the first constraint in (5.4), and as a result, the value of the objective function (5.2) (the additional criterion "min-max") does not become better than the current one (it is clear from the definition of the membership function (5.1)). In the other case (if $v_1 < v^*$), we check whether $\mid X_1\cap T \mid = 0$ or not. If so (namely, there exists no sink node included in X_1), this observation implies that the total value of any max-flow in this network is less than v^* even though capacities $c(t_j,t)$ are changed and hence this problem is infeasible.

Now we assume that $v_1 < v^*$ and $\mid X_1 \cap T \mid \neq 0$ (in Step 3 and Step 4, respectively) in order to show the validity for the updates of capacities $c(t_j, t)$ in Step 4 and Step 5. For the first iteraion $h = 1$, since $F_{\overline{X}_1}$, the total flow value to the sink nodes in $\overline{X}_1$, is less than $\sum_{t_j \in \overline{X}_1 \cap T} c(t_j, t)$, the sum of the corresponding capacities, these capacities $c(t_j, t)$ must be updated in order that their degrees of satisfaction are "perfectly balanced" (the capacities after updates are denoted by $\tilde{c}(t_j, t)$). That is, all $\mu_{t_j}(\tilde{c}(t_j, t)), t_j \in \overline{X}_1 \cap T$ have the same value $\mu'_{\overline{X}_1}$ (similarly to (5.5)) and the sum of these capacities is equal to the total value of the corresponding capacities, i.e., $\sum_{t_j \in \overline{X}_1 \cap T} \tilde{c}(t_j, t) = F_{\overline{X}_1}$. This update in Step 4 is valid in the sense of the objective function (5.2) (i.e., " max-min sharing"), since we should maximize the minimum of their degree of satisfaction if possible, although all of these degrees are smaller than the capacities before updates, i.e.,

$$\mu_{t_j}(\tilde{c}(t_j, t)) = \mu'_{\overline{X}_1} \leq \mu'_1 = c(t_j, t).$$

For the sink nodes in X_1, on the other hand, the arc capacities from these sink nodes to super sink may as well be updated so that membership values of these capacities are also perfectly balanced, i.e., all the corresponding membership values are μ'_{X_1} and the sum of their capacities is equal to $v^* - F_{\overline{X}_1}$, since the total flow value into all sink nodes must be v^*. The update in Step 4 is valid in the sense of the additional criterion (i.e., " min-max sharing"), since we should minimize the maximum of their degree of satisfaction if possible, although all of these degrees are greater than the capacities before updates, i.e.,

$$\mu_{t_j}(\tilde{c}(t_j, t)) = \mu'_{X_1} \geq \mu'_1 = c(t_j, t).$$

For the iteration $h \geq 2$, similar updates are executed, but the situation is more complicated, i.e., sink nodes are partitioned into the following four

subsets (see the left figure in Fig.5.2):

$$(1)\ \overline{X}_1 \cap \overline{X}_h,\ (2)\ \overline{X}_1 \cap X_h,\ (3)\ X_1 \cap \overline{X}_h \text{ and } (4)\ X_1 \cap X_h.$$

The total flow value to sink nodes in (1) and (2) is at most $F_{\overline{X}_1}$. In addition, it is at most $F_{\overline{X}_1 \cap \overline{X}_h}$ in case of (1) and accordingly the arc capacities from sink nodes in (1) to super sink must be updated in Step 5 so that they are perfectly balanced, i.e., all $\mu_{t_j}(\tilde{c}(t_j, t)), t_j \in \overline{X}_1 \cap \overline{X}_h \cap T$ have the same value $\mu'_{\overline{X}_1 \cap \overline{X}_h}$ (similarly to (5.5)) and the sum of their capacities is $F_{\overline{X}_1 \cap \overline{X}_h}$. For the sink nodes in (2), their capacities may as well be updated so that the corresponding membership values are perfectly balanced and the total value is $F_{\overline{X}_1} - F_{\overline{X}_1 \cap \overline{X}_h}$. Also for the sink nodes in (3) and (4), we execute similar updates (refer to the corresponding equations in Step 5).

For the operations in Step 5, it is clear that the balanced satisfaction degree with respect to sink nodes in $\overline{X}_h$, i.e., the updates of the arc capacities associated with (1) and (3) are valid in the sense of "max-min sharing" (similarly to the case of $\overline{X}_1 \cap T$ at $h = 1$). On the other hand, for the updates of those associated with X_h, i.e., (2) and (4), keeping the balance of the corresponding degrees is meaningless in terms of "max-min sharing", since the membership values for the updated capacities $\tilde{c}(t_j, t)$ are not less than those for the capacities before updates, i.e.,

$$\mu_{t_j}(\tilde{c}(t_j, t)) = \mu'_{X_1 \cap X_h} \geq \mu_{t_j}(c(t_j, t)),$$

where $\tilde{c}(t_j, t)$ denote the capacities to which updated from $c(t_j, t)$ at the $h-$th iteration. However it is meaningful from a different point of view, namely "min-max sharing" (similarly to the case of $X_1 \cap T$ at $h = 1$), since the membership values with respect to these sink nodes may as well be minimized. Therefore this algorithm takes account of not only max-min sharing but also min-max sharing (we call it *optimal sharing* [Ich]).

Remark 5.1 Algorithm 5.1 attains optimal sharing.

Next let us discuss the complexity of Algorithm 5.1.

Lemma 5.1 *The node set* $\overline{X}_1 \cap \overline{X}_h \cap T$ *includes* $\overline{X}_1 \cap \overline{X}_h \cap \overline{X}_{h+1} \cap T$ *as a proper subset.*

Proof. It is sufficient to show that $\overline{X}_1 \cap \overline{X}_h \cap T \neq \overline{X}_1 \cap \overline{X}_h \cap \overline{X}_{h+1} \cap T$. For this, assume that equality holds here. We consider the time instant at which the $h-$th iteration of Algorithm 5.1 has just finished, i.e., the arc capacities associated with $\overline{X}_1 \cap \overline{X}_h \cap T$ has been updated so that $\sum_{t_j \in \overline{X}_1 \cap \overline{X}_h \cap T} c(t_j, t) = F_{\overline{X}_1 \cap \overline{X}_h}$ holds. Then assume that the total flow value into the sink nodes in $\overline{X}_1 \cap \overline{X}_h \cap T$ satisfies $\sum_{t_j \in \overline{X}_1 \cap \overline{X}_h \cap T} f(t_j) < F_{\overline{X}_1 \cap \overline{X}_h}$ after the $(h+1)$-st iteration (otherwise, the algorithm has already terminated at the $h-$th iteration). Since $\overline{X}_1 \cap \overline{X}_h \cap T = \overline{X}_1 \cap \overline{X}_h \cap \overline{X}_{h+1} \cap T$ (i.e., $\overline{X}_h \subseteq \overline{X}_{h+1}$), this implies that $f < \sum_{t_j \in \overline{X}_1 \cap \overline{X}_h \cap T} c(t_j, t)$, where f denotes the capacity of the minimum cut separating s from the sink nodes in $\overline{X}_1 \cap \overline{X}_h$ after the $(h+1)-$st iteration (this is shown in Fig.5.2). However it is clear from max-flow min-cut theorem (refer to Subsection 1.4.2) that $F_{\overline{X}_1 \cap \overline{X}_h} = f$, since the value of every arc capacity except $c(t_j, t)$ is not changed, and we have a contradiction $f = F_{\overline{X}_1 \cap \overline{X}_h} < \sum_{t_j \in \overline{X}_1 \cap \overline{X}_h \cap T} c(t_j, t) = F_{\overline{X}_1 \cap \overline{X}_h}$. Consequently the set $\overline{X}_1 \cap \overline{X}_h \cap T$ is not equal to $\overline{X}_1 \cap \overline{X}_h \cap \overline{X}_{h+1} \cap T$. □

Similarly, the following lemma also holds.

Lemma 5.2 *The node set of* $X_1 \cap \overline{X}_h \cap T$ *includes* $X_1 \cap \overline{X}_h \cap \overline{X}_{h+1} \cap T$ *as a proper subset.*

From these lemmas, it is clear that Algorithm 5.1 repeats the computations of maximum flow at most l $(=\mid T \mid)$ times, establishing the following theorem.

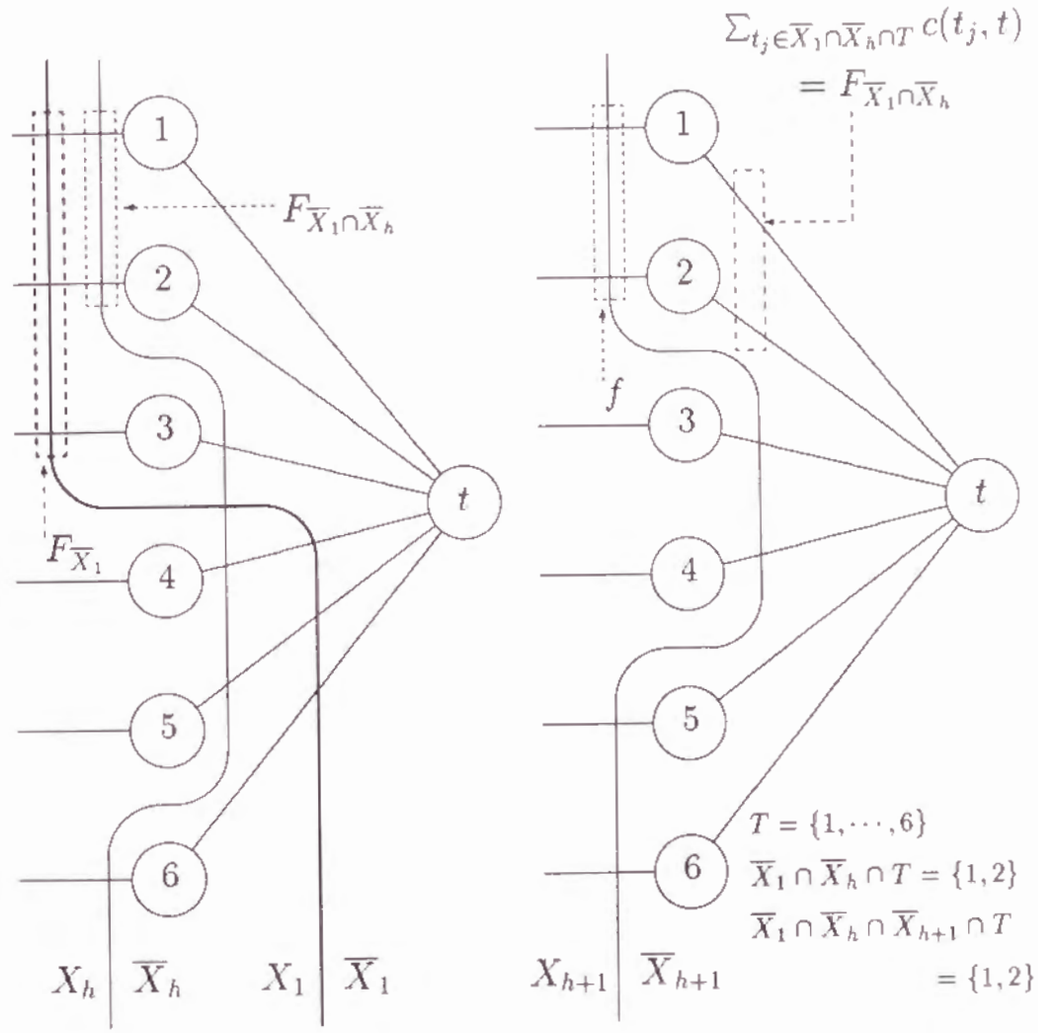

Fig. 5.2 Example of the minimum cuts
(Left: the h-th iteration, Right: the $(h+1)$-st iteration)

Theorem 5.1 *Algorithm 5.1 for solving the fuzzy sharing problem finds an optimal solution in $O(l\,M(n,m))$ computational time where $l =\mid T \mid$ and $M(n,m)$ denotes the time bound for finding the maximum flow when $n =\mid N' \mid$ and $m =\mid A' \mid$.*

Proof. It is clear from the fact that the time complexity for finding the maximum flow dominates that for updating (setting) the capacities from sink nodes t_j to a super sink t in Step 4 and Step 5 (in Step 1). □

Example 5.1

In the rest of this section, we consider an numerical example of $P8$ for the network shown in Fig.5.3, where the set of source nodes is $S = \{1,2\}$ and that of sink nodes is $T = \{4,5,6\}$ (for simplicity of presentation, two source nodes $\{s_1, s_2\}$ and three sink nodes $\{t_1, t_2, t_3\}$ are simply written as $\{1,2\}$ and $\{4,5,6\}$, respectively). The capacity is attached to each arc in the figure. Now it is assumed that v^*, the total amount of resource to be shared, is 18. The membership functions of sink nodes are given in Fig.5.4, i.e.,

$$\mu_4(f(4)) = \begin{cases} 0 & \text{if } f(4) \le 0, \\ \dfrac{f(4)}{10} & \text{if } 0 < f(4) < 10, \\ 1 & \text{if } f(4) \ge 10, \end{cases}$$

$$\mu_5(f(5)) = \begin{cases} 0 & \text{if } f(5) \le 2, \\ \dfrac{f(5)-2}{3} & \text{if } 2 < f(5) < 5, \\ 1 & \text{if } f(5) \ge 5, \end{cases}$$

$$\mu_6(f(6)) = \begin{cases} 0 & \text{if } f(6) \le 2, \\ \dfrac{f(6)-2}{5} & \text{if } 2 < f(6) < 7, \\ 1 & \text{if } f(6) \ge 7. \end{cases}$$

The 1-st iteration.

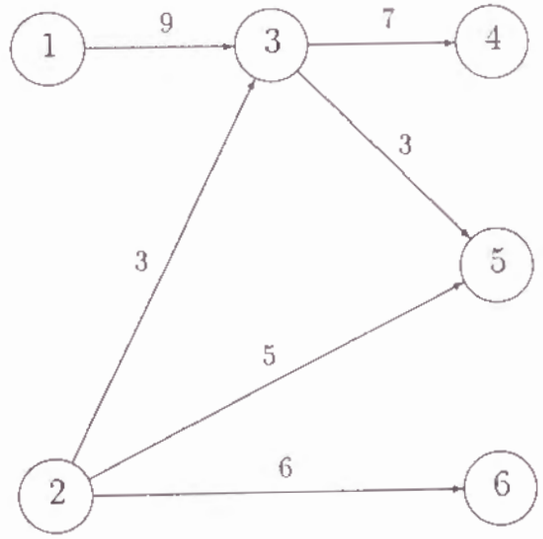

Fig. 5.3 The network for an illustrative example

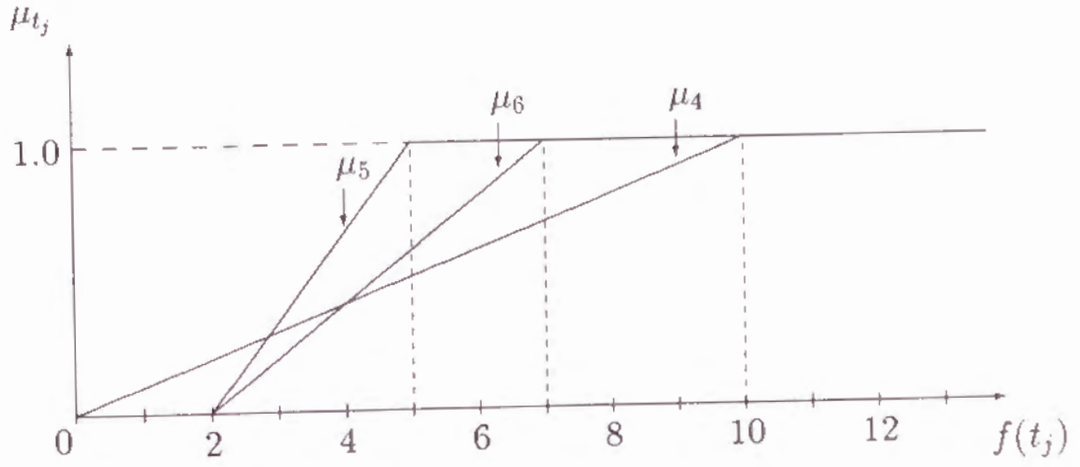

Fig. 5.4 Membership functions $\mu_{t_j}(f(t_j))$

Step 1. After setting that $h = 1$ and $c(s,1) = c(s,2) = \infty$, the value of μ_1' is calculated as follows:

$$\mu_1' = (18 - (0 + 2 + 2))/((10 - 0) + (5 - 2) + (7 - 2)) = 7/9.$$

Since $\mu_1' = 7/9 \not\leq 0$, set

$$c(4,t) = (7/9)(10 - 0) + 0 = 70/9,$$
$$c(5,t) = (7/9)(5 - 2) + 2 = 39/9,$$
$$c(6,t) = (7/9)(7 - 2) + 2 = 53/9.$$

Step 2. The result of the max-flow computation as well as its minimum cut $(X_1, \overline{X}_1)$ is shown in Fig.5.5, where the first number in parentheses beside each arc indicates the value of its flow, and the second number indicates its capacity.

Step 3. Since $v_1 = 155/9 < 18 = v^*$, go to Step 4.

Step 4. The current situation is as follows: $h = 1$, $X_1 \cap T = \{5,6\}$ and $\overline{X}_1 \cap T = \{4\}$ (see Fig.5.5). Accordingly, it follows that $F_{\overline{X}_1} = f(4) = 7$ and

$$\mu_{\overline{X}_1}' = (7 - 0)/(10 - 0) = 7/10,$$
$$\mu_{X_1}' = (18 - 7 - (2 + 2))/((5 - 2) + (7 - 2)) = 7/8.$$

By using these data, the new capacities from sink nodes to super sink are set as follows:

$$c(4,t) = (7/10)(10 - 0) + 0 = 7,$$
$$c(5,t) = (7/8)(5 - 2) + 2 = 37/8,$$
$$c(6,t) = (7/8)(7 - 2) + 2 = 51/8.$$

Finally, set $h = 2$ and return to Step 2.

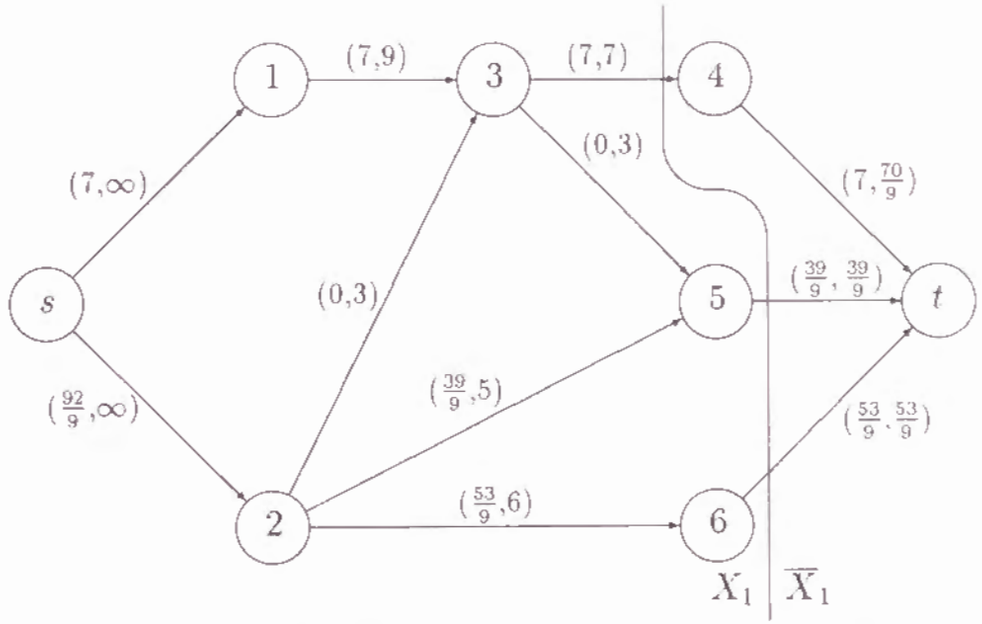

Fig. 5.5 The result of the max-flow computation at $h = 1$

The 2-nd iteration.

Step 2. The result of the max-flow computation and its minimum cut $(X_2, \overline{X}_2)$ are shown in Fig.5.6.

Step 3. Since $v_2 = 141/8 < 18 = v^*$, go to Step 4. Then we move to Step 5 immediately since h=2.

Step 5. Since $\overline{X}_1 \cap \overline{X}_2 \cap T = \{4\}$, $X_1 \cap \overline{X}_2 \cap T = \{6\}$ and $X_1 \cap X_2 \cap T = \{5\}$ (see Fig.5.6), we obtain $F_{\overline{X}_1 \cap \overline{X}_2} = f(4) = 7$ and $F_{X_1 \cap \overline{X}_2} = f(6) = 6$. Accordingly,

$$\mu_{\overline{X}_1 \cap \overline{X}_2}' = (7 - 0)/(10 - 0) = 7/10,$$
$$\mu_{X_1 \cap \overline{X}_2}' = (6 - 2)/(7 - 2) = 4/5,$$
$$\mu_{X_1 \cap X_2}' = (18 - 7 - 6 - 2)/(5 - 2) = 1,$$

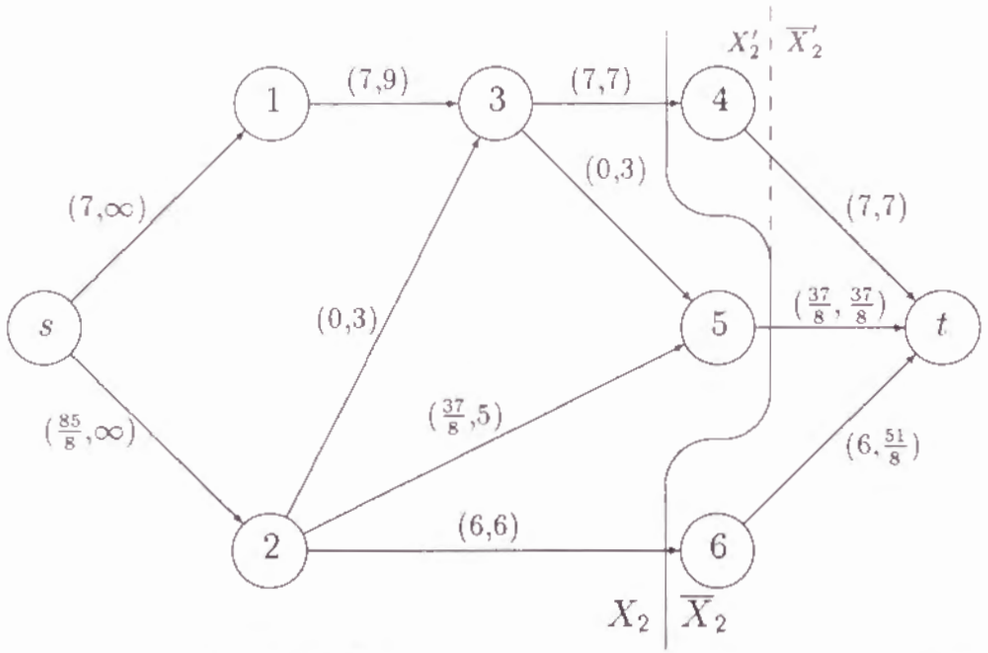

Fig. 5.6 The result of the max-flow computation at $h = 2$

and the capacities are updated as follows:

$$c(4,t) = (7/10)(10-0)+0 = 7,$$

$$c(5,t) = 1(5-2)+2 = 5,$$

$$c(6,t) = (4/5)(7-2)+2 = 6.$$

Note that although there are two different minimum cuts ($c(X_2, \overline{X}_2) = c(X_2', \overline{X}_2') = 141/8 = v_2$; see Fig.5.6), the values of these capacities are the same even if the calculation in case of cut $(X_2', \overline{X}_2')$ is executed. After setting $h = 3$, return to Step 2.

The 3-rd iteration.

Step 2. The result of the max-flow computation at this iteration is shown in Fig.5.7.

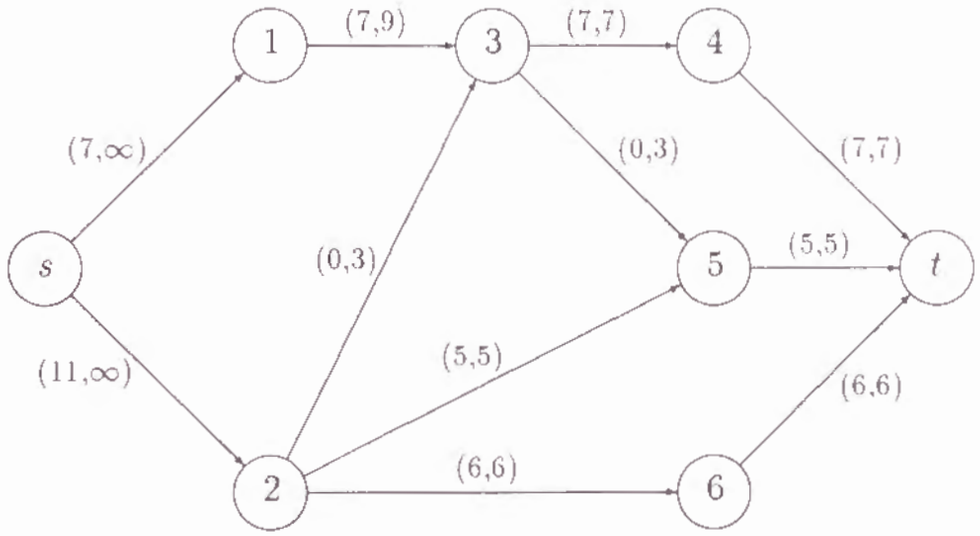

Fig. 5.7 The result of the max-flow computation at $h = 3$

Step 3. Since $v_3 = 18 = v^*$, the current flow is optimal and the algorithm halts.

The degrees of satisfaction with respect to the current share are $\mu_4(7) = 0.7, \mu_5(5) = 1$ and $\mu_6(6) = 0.8$, and therefore the optimal max-min value of the objective funcion (5.2) is 0.7 (in addition, the optimal min-max value is 1).

5.3 Generalized fuzzy sharing problem

5.3.1 Formnlation and solution procedure

In some situation, sharing may be done under the constraint that the recieved amount of resource at each sink node is a multiple of some "block-unit" (e.g., a dozen). To incorporate this kind of constraint, we consider a generalized fuzzy sharing problem in which the recieved amount must be a multiple of

a certain positive integer d (we call it $d-multiple$). In this subsection, we formulate this problem as $P9$ and give a polynomial time algorithm. The validity and complexity of the algorithm will be given in the next subsection.

$$P9: \quad \text{Maximize} \quad \min_{t_j \in T} \mu_{t_j}(f(t_j)) \tag{5.7}$$

subject to

$$\sum_{y \in A(x)} f(x,y) - \sum_{y \in B(x)} f(y,x) = \begin{cases} v^* & \text{if } x = s \\ 0 & \text{if } x \neq s,t \\ -v^* & \text{if } x = t, \end{cases}$$

$$0 \leq f(x,y) \leq c(x,y), \ x \in N' - \{t\}, y \in N' - \{s\}.$$

$$f(t_j) = kd \quad \text{for some positive integer } k, \tag{5.8}$$

where d is a given positive integer. This problem is the same as $P8$ except that the $d-$multiple constraint is added (namely, $P9$ is "max-min sharing" with the $d-$multiple constraint; we will discuss the comparison of this "max-min" version and the other "min-max" in Remark 5.2 and Example 5.2). In the following algorithm, we use the notation "a mod b". This outputs the residue (remainder) of a divided by b, e.g., 17 mod 6 outputs 5. Moreover, when we find a maximum flow at the $h-$th iteration, we use the notation $c_h(t_j,t)$ which denotes the arc capacity from sink t_j to super sink t in the network.

Now we outline Algorithm 5.2 to solve $P9$ (its validity and complexity will be discussed in Subsection 5.3.2). In Step 1 of the algorithm, we utilize the optimal flow to the previous problem $P8$ by using Algorithm 5.1. That is, if the flow value $f(t_j)$ to every sink node t_j satisfies the $d-$multiple constraint, the algorithm terminates (the current flow is optimal; for details, see Subsection 5.3.2). Otherwise, capacities $c(t_j,t)$ must be updated in order that every $f(t_j)$ satisfies the constraint. Accordingly, we update the capacities to $c_h(t_j,t) = d\lfloor c_0(t_j,t)/d \rfloor$, $t_j \in T$ in Step 1 at the first iteration $h = 1$, where $c_0(t_j,t)$ denote the capacities from t_j to t in the final network realized by Algorithm

5.1. In Step 2, we find a maximum flow in the updated network. Since v_1 (the total value of the max-flow at the first iteration) is less than v^* (see Subsection 5.3.2), we update capacities $c(t_j,t)$ further (in Step 4) while taking account of the view point of max-min sharing (i.e., the objective of $P9$). At this time, we find an augmenting path (max-flow) from s to t via t_j when $c_h(t_j,t)$ is updated to $c_{h+1}(t_j,t) = c_h(t_j,t) + d$ for every sink $t_j \in T$ (i.e., the computation of max-flow is executed l times an iteration) in order to narrow the candidates (to be updated) $t_j \in T$ down to $t_j \in A_1 \subseteq T$. Here the notation A_1 (A_h) denotes the set of the sink nodes t_j possible to increase the flow values $f(t_j)$ by d as a result of the augmentation at the first ($h-$th) iteration. That is, after we select a sink node t'_j which has the lowest degree of satisfaction among the sink nodes in A_1 (A_h) at the first ($h-$th) iteration and the capacity $c(t'_j,t)$ is increased by d, we find a max-flow in Step 2 at the second ($(h+1)-$st) iteration. Note that $f(t'_j) = c_h(t'_j,t) + d$ holds without fail at the $(h+1)-$st iteration, since we check the realization by executing the augmentation in advance (at the $h-$th iteration). Except for Step 1, each step is repeatedly executed until the total value of max-flow is equal to v^*.

Algorithm 5.2

Step 1. By using Algorithm 5.1, solve the fuzzy sharing problem. If $f(t_j)$ mod $d = 0$ for all $t_j \in T$ then stop (i.e., the current flow is optimal). Otherwise set $h := 1$ and $c_h(t_j,t) := d\lfloor c_0(t_j,t)/d \rfloor$, $t_j \in T$, where $c_0(t_j,t)$ denotes the capacity in the final network realized by Algorithm 5.1.

Step 2. Find a maximum flow f_h from s to t and its value v_h.

Step 3. If $v_h < v^*$ then go to Step 4. Otherwise stop; the current flow is optimal.

Step 4. Find a sink node t'_j such that $\mu_{t'_j}(c_h(t'_j,t)) = \min_{t_j \in A_h} \mu_{t_j}(c_h(t_j,t))$, where A_h is the set of t_j such that there exists an augmenting path from s to t via t_j when $c_h(t_j,t)$ for every $t_j \in T$ is updated to $c_{h+1}(t_j,t) := c_h(t_j,t) + d$. And, for t'_j, set $c_{h+1}(t'_j,t) := c_h(t'_j,t) + d$. If A_h is empty then stop; this problem is infeasible.

Step 5. Set $h := h + 1$ and return to Step 2.

5.3.2 Validity and complexity of Algorithm 5.2

First we show the validity of the Algorithm 5.2. Assume that $f(t_j) \bmod d \neq 0$ for at least one sink node in Step 1 (at the first iteration $h = 1$) of the algorithm; otherwise the algorithm terminates since the current flow is optimal, that is, every $f(t_j)$ (flow value to sink node t_j) of the resultant flow realized by Algorithm 5.1 which attains "max-min sharing" satisfies the condition of $d-$multiple constraint. After the max-flow computation (in Step 2) at the first iteration (let $(X_1, \overline{X}_1)$, $s \in X_1, t \in \overline{X}_1$ be its corresponding minimum cut), each $f(t_j)$ satisfies the condition of $d-$multiple constraint and

$$f(t_j) = c_1(t_j,t) = d\lfloor c_0(t_j,t)/d \rfloor,$$

since $c_1(t_j,t) \leq c_0(t_j,t)$ and in addition, the minimum cut of the final max-flow derived from Algorithm 5.1 (let $(X_0, \overline{X}_0)$, $s \in X_0, t \in \overline{X}_0$ be the minimum cut) shows that all sink nodes t_j are included in X_0 and there exists only super sink t in $\overline{X}_0$ (it is clear from max-flow min-cut theorem). Therefore, although the present ($h = 1$ in Algorithm 5.2) degree of satisfaction with respect to every sink t_j is not greater than that realized by Algorithm 5.1, all sink nodes are included in X_1. This implies there is no surplus and deficit in the flow required at the present iteration ($h = 1$), namely, $\sum_{t_j \in T} f(t_j) = \sum_{t_j \in T} c_1(t_j,t)$.

From the above assumption that there exists at least one sink which does not satisfy the constraint of $d-$multiple, $v_1 < v^*$ holds. Accordingly we must determine a sink node t'_j (in Step 4) to augment flow by d. The sink node t'_j which has the worst degree of satisfaction at the current iteration $h = 1$ should be selected in the sense of "max-min sharing". If the sink node was not selected from A_1 that consists of the sink nodes t_j for which there exists an augmenting path from s to t via t_j when we increase its capacity by d, the total flow into the sink node ($\notin A_1$) would be less than the required value $(c_1(t'_j,t) + d)$ at the next iteration ($h = 2$). Consequently the sink node t'_j (in Step 4) with a revised capacity necessarily belongs to A_1. In case of $h \geq 2$, the capacity from the sink node t'_j to super sink t is updated in the same manner.

Next we give some lemmas necessary for showing the complexity of Algorithm 5.2.

Lemma 5.3 $v^* - v_1 \leq (d-1) \mid T \mid$.

Proof. Let $f^*(t_j)$ be the total value of the optimal flow to sink node t_j when the $d-$multiple constraint is deleted. Then

$$\begin{aligned} v^* - v_1 &= \sum_{t_j \in T} f^*(t_j) - \left(\sum_{t_j \in T} d\lfloor c_0(t_j,t)/d \rfloor \right) \\ &\leq \sum_{t_j \in T} (d-1) \\ &= (d-1) \mid T \mid . \quad \square \end{aligned}$$

Lemma 5.4 *If* $v_h < v^*$, *then* $v_{h+1} = v_h + d$.

Proof. It is clear from the definition of A_h. $\square$

Lemma 5.5 *If* $v_h < v^*$, *then* $A_{h+1} \subseteq A_h$.

Proof. It is clear from the definition of A_h and Lemma 5.4. $\square$

Theorem 5.2 *The time complexity of Algorithm* 5.2 *is* $O(l^2M(n,m))$, *where* $l =\mid T \mid$ *and* $M(n,m)$ *denotes the time bound for finding the maximum flow when* $n =\mid N' \mid, m =\mid A' \mid$.

Proof. It is clear from Theorem 5.1 that Step 1 takes at most $O(l\,M(n,m))$ time. The time required for each execution of Step 2 to Step 5 is as follows.

Step 2 : $O(M(n,m))$.

Step 3 : $O(1)$.

Step 4 : $O(l\,M(n,m))$ to find l augmenting paths.

Step 5 : $O(1)$.

From three lemmas (Lemma 5.3, 5.4 and 5.5), it is clear that the algorithm terminates in at most $l(d-1)/d$ iterations. Therefore, Algorithm 5.2 finds an optimal solution for $P9$ in $O(l^2M(n,m))$ computational time. □

Remark 5.2 If the objective function of $P9$ is changed to "minimize $\max_{t_j} \mu_{t_j}$", we modify Step 4 of Algorithm 5.2 as follows:

> Set $c_{h+1}(t'_j,t) = c_h(t'_j,t) + d$, where t'_j is the sink node that gives $\min_{t_j \in A_h} \mu_{t_j}(c_h(t_j,t)+d)$.

From Remark 5.1, Algorithm 5.1 used in Step 1 of Algorithm 5.2 is also valid for the min-max sharing. In addition, for this modification, it is valid in the sense of the min-max sharing that selecting a sink node with the worst degree of satisfaction as t'_j after each total flow value to all sink nodes in A_h is increased by d. Note that the optimal flow for the min-max sharing is in general different from that for the max-min sharing (see the following example).

Example 5.2

Now we illustrate the behavior of Algorithm 5.2 for solving $P9$ (the general fuzzy sharing problem) on the same network as shown in Fig.5.3. To formulate

the problem, we assume that the "block unit" is 3. That is, we obtain an optimal flow that is 3-multiple.

The 1-st iteration.

Step 1. Note that

$$f(4) \bmod 3 = 7 \bmod 3 = 1,$$
$$f(5) \bmod 3 = 5 \bmod 3 = 2,$$
$$f(6) \bmod 3 = 6 \bmod 3 = 0,$$

holds. Hence set $h = 1$ and

$$c_1(4,t) = 3\lfloor 7/3 \rfloor = 6,\ c_1(5,t) = 3\lfloor 5/3 \rfloor = 3,\ c_1(6,t) = 3\lfloor 6/3 \rfloor = 6.$$

Step 2. The result of the max-flow computation is shown in Fig.5.8.

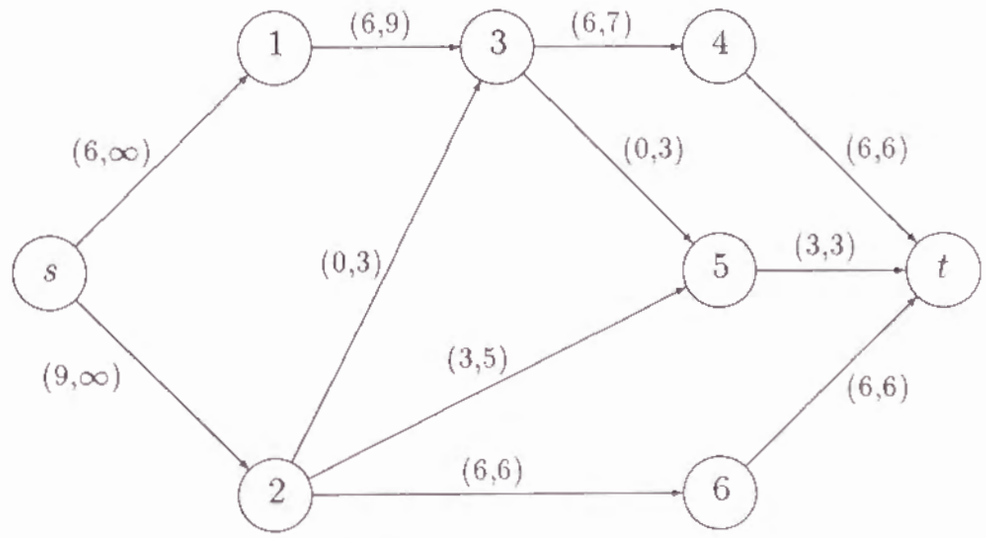

Fig. 5.8 The result of the max-flow computation at $h = 1$

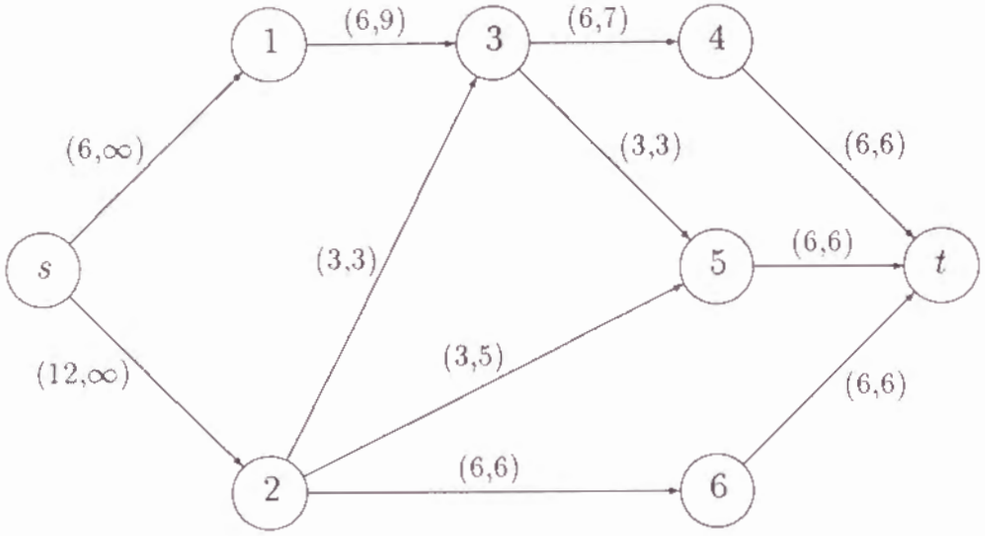

Fig. 5.9 The result of the max-flow computation at $h = 2$

Step 3. Since $v_1 = 15 < 18 = v^*$, go to Step 4.

Step 4. Since $A_1 = \{5\}$, set $c_2(5,t) = c_1(5,t) + 3 = 6$ without checking that $\min_{t_j \in A_1} \mu_{t_j}(c_1(t_j,t))$.

Step 5. After setting $h = 2$, return to Step 2.

At the second iteration, we obtain an optimal flow such that $\mu_4(6) = 0.6, \mu_5(6) = 1$ and $\mu_6(6) = 0.8$ (see Fig.5.9). Therefore the optimal value is 0.6.

In the above example, even if the "max-min" operation in (5.7) is replaced by "min-max", the optimal solution does not change (since, in Step 4, we can no longer increase the flow to sink node "4" nor "6" by 3-unit). However, for the other network obtained by changing the capacity of arc $(3,4)$ to $c(3,4) = 9$, there is a difference between the max-min and min-max solutions. In this case, we have $A_1 = \{4,5\}$ in Step 4 at the first iteration of Algorithm 5.2. Then,

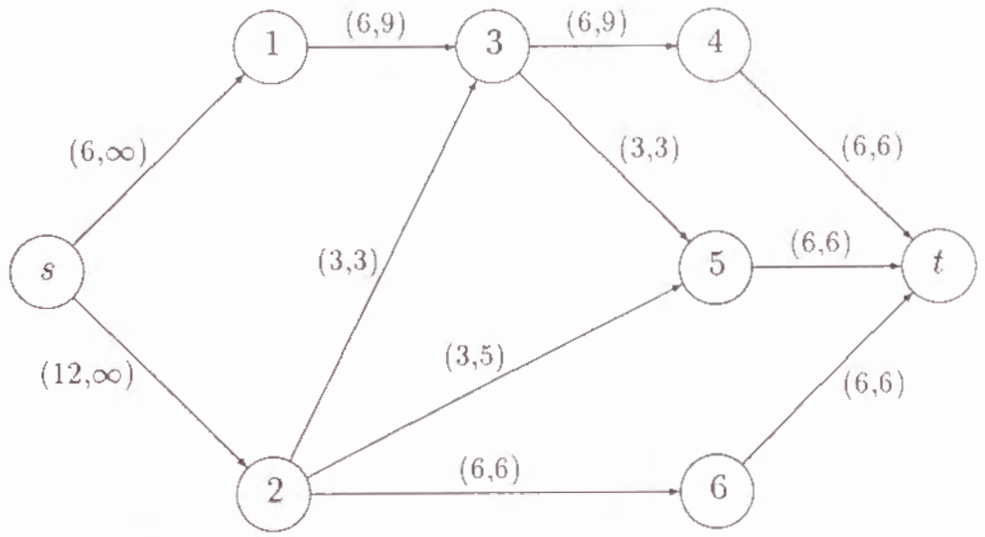

Fig. 5.10 The result of min-max version of $P9$

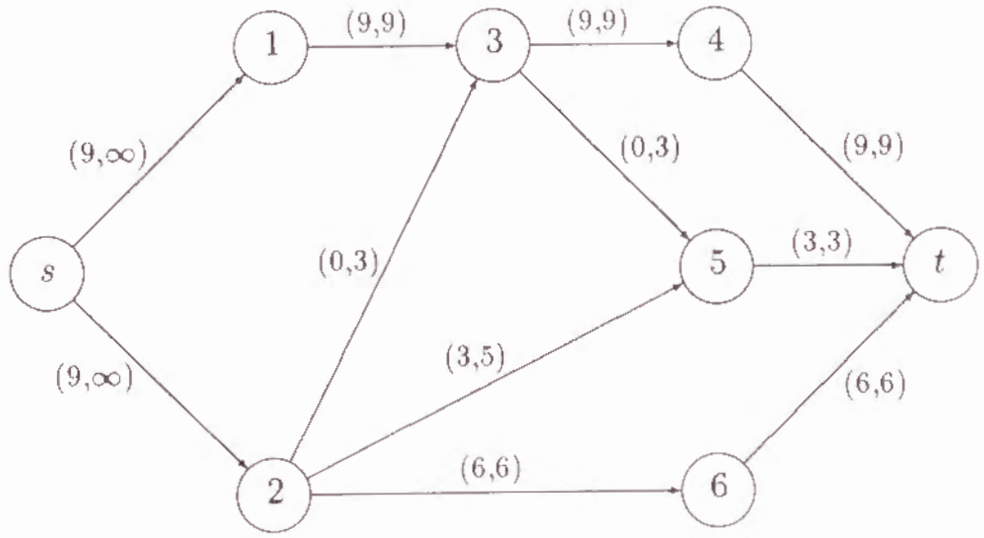

Fig. 5.11 The result of max-min version of $P9$

$$\mu_4(c_1(4,t)) = \mu_4(6) = 0.6,\ \mu_5(c_1(5,t)) = \mu_5(3) = 0.33,$$

and hence $t' = 5$. On the other hand, in the case of "min-max" version, the modifed algorithm gives another solution, i.e., $t' = 4$, since

$$\mu_4(c_1(4,t)+d) = \mu_4(6+3) = 0.9,\ \mu_5(c_1(5,t)+d) = \mu_5(3+3) = 1.$$

The results of these "max-min" and "min-max" versions are illustrated in Fig.5.10 and Fig.5.11, respectively.

Chapter 6

FUZZY TRANSPORTATION PROBLEM

6.1 Introduction

The transportation problem described briefly in Subsection 1.4.3 (Paragraph B) is often represented by a bipartite network that consists of two node-sets, i.e., sets of supply (or plant) and demand (or warehouse) nodes, respectively. The problem is to determine a flow such that the total transportation cost is minimized. In order to consider the real situation in which it may not be easy to determine the amounts of supplies and demands as crisp numbers, we consider a fuzzy version of the transportation by introducing two kinds of membership functions corresponding to supplies and demands. Each membership function describes the degree of satisfaction for the flow values sent from a supply node or sent to a demand node. We assume that these membership functions are determined a priori by DM (a decision-maker). The objective of our problem is to determine an optimal flow that maximizes the smallest value of all membership functions (i.e., a determination of equitable distribution) under the constraint that the total transportation cost must not exceed a certain upper limit.

In the next section (Section 6.2), we formulate the problem and derive some properties of the problem. After that, the idea of the proposed solution procedure is first informally described and then a polynomial time algorithm is presented. In Section 6.3, we explore another fuzzy transportation problem in which the constraint of integral flow is added.

6.2 Fuzzy transportation problem

6.2.1 Problem formulation

We consider the fuzzy transportation problem on a bipartite network $BG = [S, T; \mathcal{A}]$, where S and T denote sets of supply nodes and demand nodes, respectively, and $\mathcal{A}$ denotes the set of arcs from S to T. Unlike the fuzzy sharing problems discussed in Chapter 5, there are two kinds of membership functions characterizing the satisfaction degrees of not only demand nodes (sinks) but also supply nodes (sources). The total amount of supply from $s_i \in S$ and that of demand at $t_j \in T$ are denoted by f_{s_i} and f_{t_j}, respectively. Note that these f_{s_i} and f_{t_j} are not constants but variables to be determined. Now two kinds of membership functions $\mu_{\boldsymbol{A}s_i}(f_{s_i})$ (for fuzzy supplies $\boldsymbol{A}s_i$) and $\mu_{\boldsymbol{A}t_j}(f_{t_j})$ (for fuzzy demands $\boldsymbol{A}t_j$), which are abbreviated to $\mu_{s_i}(f_{s_i})$ and $\mu_{t_j}(f_{t_j})$ respectively, characterize the degrees of satisfaction at supplies and demands (see Figs. 6.1 and 6.2):

$$\mu_{s_i}(f_{s_i}) = \begin{cases} 1 & \text{if } f_{s_i} \le a_{s_i}, \\ -\dfrac{f_{s_i} - b_{s_i}}{b_{s_i} - a_{s_i}} \triangleq F_{s_i}(f_{s_i}) & \text{if } a_{s_i} < f_{s_i} < b_{s_i}, \\ 0 & \text{if } f_{s_i} \ge b_{s_i}, \end{cases} \tag{6.1}$$

$$\mu_{t_j}(f_{t_j}) = \begin{cases} 0 & \text{if } f_{t_j} \le d_j, \\ \dfrac{f_{t_j} - d_{t_j}}{e_{t_j} - d_{t_j}} \triangleq G_{t_j}(f_{t_j}) & \text{if } d_{t_j} < f_{t_j} < e_{t_j}, \\ 1 & \text{if } f_{t_j} \ge e_{t_j}, \end{cases} \tag{6.2}$$

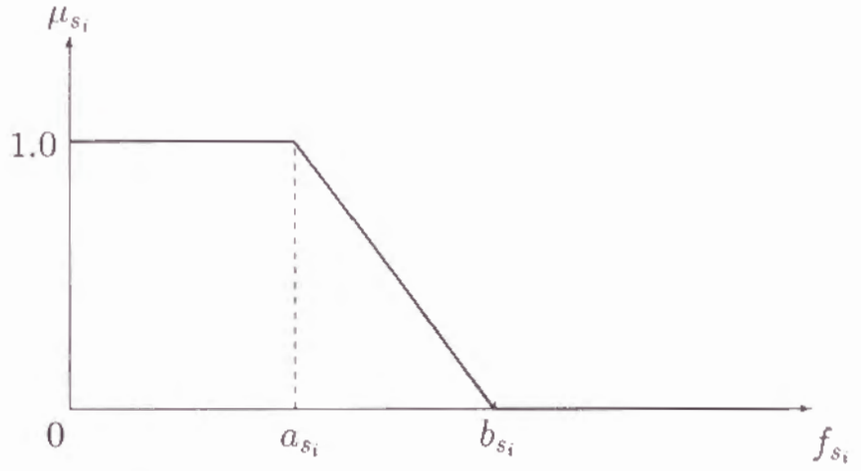

Fig. 6.1 Membership function $\mu_{s_i}(f_{s_i})$ for supply node s_i

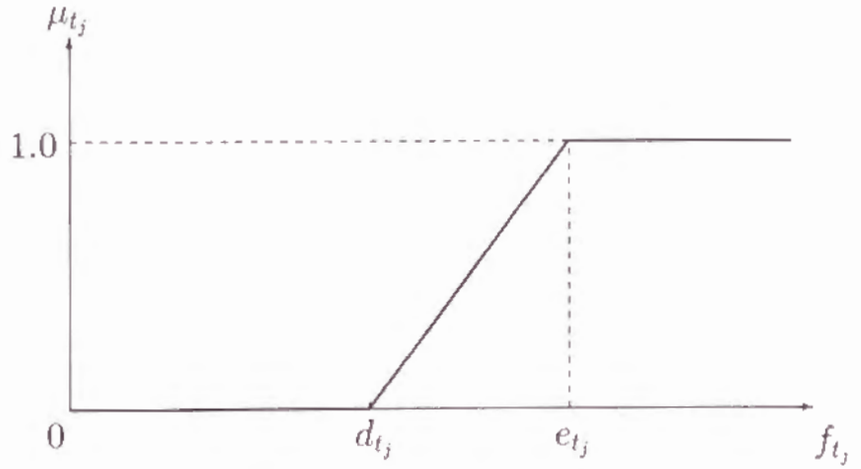

Fig. 6.2 Membership function $\mu_{t_j}(f_{t_j})$ for demand node t_j

That is, for a supply node s_i, it is possible to send out some stocks not greater than a_{s_i} unconditionally (i.e., $\mu_{s_i} = 1$), but if the amount of supply exceeds a_{s_i}, the degree of satisfaction decreases (i.e., $0 < \mu_{s_i} < 1$) according to its overflow quantity because the supply node faces the "out of stock" situation, and it is impossible to manage to provide the amount greater than b_{s_i} (i.e., $\mu_{s_i} = 0$). On the other hand, the membership funcion μ_{t_j} for a demand node t_j is determined in the same manner as that for a sink node in the fuzzy sharing problem in Chapter 5.

Each arc (s_i, t_j) links a supply node s_i to a demand node t_j and the transportation cost c_{s_i,t_j} per unit flow is associated with arc (s_i, t_j). While the ordinary transportation problem minimizes the total cost, we formulate its fuzzy version in the following manner.

$$P10: \quad \text{Maximize} \quad \min\left\{\mu_{s_i}(f_{s_i}), \mu_{t_j}(f_{t_j}) \mid s_i \in S, t_j \in T\right\}, \tag{6.3}$$

$$\text{subject to} \quad \sum_{t_j \in T} f(s_i, t_j) = f_{s_i}, \quad i = 1, \cdots, m, \quad \sum_{s_j \in S} f(s_i, t_j) = f_{t_j}, \quad j = 1, \cdots, n, \tag{6.4}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_{s_i,t_j} f(s_i, t_j) \leq \overline{C}, \tag{6.5}$$

$$f(s_i, t_j) \geq 0, \; s_i \in S, t_j \in T, \tag{6.6}$$

where $\overline{C}$ is a given upper bound on the total transportation cost. Then we assume $\sum_{s_i \in S} b_{s_i} > \sum_{t_j \in T} d_{t_j}$ in (6.1) and (6.2), since otherwise any feasible solution has the objective value 0.

6.2.2 Preliminaries to solving $P10$

In this subsection, we discuss some procedures and remark on solving $P10$ (a solution algorithm for $P10$ will be presented in the next subsection). The objective of $P10$ is to maximize the smallest value of all membership fuctions

under the upper bound (6.5). The best equiatable assignment is attained in the sense of "max-min", if all membership functions have a same value. In this case, we call it a *perfect assignment*. Once the values of f_{s_i} and f_{t_j} such that $\sum_i f_{s_i} = \sum_j f_{t_j}$ are determined, the problem is reduced to the ordinary transpotation problem. Consequently, given the flow values f_{s_i} and f_{t_j} that realize a perfect assignment, we can determine a minimum cost flow from S to T by using an algorithm for the ordinary transportation problem (e.g., Munkres' method [Mun]), and we check its feasibility of the total cost constraint (6.5). If it is feasible, the current flow is optimal. Otherwise we must modify the values of f_{s_i} and f_{t_j} in order to make its total cost less than the current one. This modification will then be repeated until the corresponding minimum cost flow satisfies (6.5). Hence we concentrate on the problem of finding the optimal value of supply and demand that maximizes the greatest value of (6.4) among feasible solutions.

First, we consider problem $P10$ without constraint (6.5). The following procedure is used to seek the optimal values of f_{s_i} and f_{t_j} for this problem (for its numerical example; see Step 0 of The 1-st iteration in Subsection 6.2.4, because we use this procedure for determining an initial solution at Step 0 in the algorithm to solve $P10$).

Procedure 6.1

Step 1. Set all degrees of membership functions as parameter α_{PA} and solve for f_{s_i} and f_{t_j}, namely find the following equations:

$$f_{s_i} = F_{s_i}^{-1}(\alpha_{PA}), \; f_{t_j} = G_{t_j}^{-1}(\alpha_{PA}). \tag{6.7}$$

Step 2. The value of α_{PA} is uniquely determined from

$$\sum_{s_i \in S} F_{s_i}^{-1}(\alpha_{PA}) = \sum_{t_j \in T} G_{t_j}^{-1}(\alpha_{PA}), \tag{6.8}$$

and the optimal values of f_{s_i} and f_{t_j} that realize a perfect assignment are obtained by substituing this α_{PA} into (6.7) in Step 1.

Remark 6.1 Although Procedure 6.1 does not take account of the upper bound constraint (6.5), it always attains a perfect assignment.

Next, we consider the upper bound (6.5). If the total cost of minimum cost flow derived from the f_{s_i} and f_{t_j} of Procedure 6.1 exceeds the upper bound (6.5), the current values of f_{s_i} and f_{t_j} must be modified. Naturally, this modification decreases the total flow value, and as a result, the satisfaction degrees at the supply side increase but those at the demand side decrease (this is clear from the definitions (6.1) and (6.2)). Accordingly, this modification should be conducted in the max-min manner, i.e., the transportation volume should be reduced while uniformly maintaining the degrees of satisfaction at the demand side. At this time, since the satisfaction degrees of the demand nodes should be maintained as high as possible, it may be better to find the arc $(\hat{s}_i, t_j)$ adjacent to each demand node t_j with the greatest unit flow cost $c_{\hat{s}_i,t_j}$ among the arcs used in the current flows, namely

$$A_j = \{(s_i, t_j) \mid f(s_i, t_j) > 0, i = 1, \cdots, m\}, \ j = 1, \cdots, n,$$
$$c_{\hat{s}_i,t_j} = \max_{(s_i,t_j)\in A_j} c_{s_i,t_j}, \ j = 1, \cdots, n,$$

and reduce the flow in these arcs $(\hat{s}_i, t_j), j = 1, \cdots, n$, while keeping the degrees of satisfaction of all demand nodes equal, until the flow in at least one arc among the n arcs $(\hat{s}_i, t_j)$ becomes 0. Then, after calculating the new total transportation cost for these revised data, check whether the current flow satisfies the cost constraint or not. If not, i.e., $\sum_{i,j} c_{s_i,t_j} f(s_i, t_j) > \overline{C}$, then the above operation is repeated.

Now we formally describe the above operation as the following procedure, in which $f_{t_j}^{(h)}$ denotes the value of f_{t_j} after the $h-$th iteration and these $f_{t_j}^{(h)}$ satisfy

$$\mu_{t_1}(f_{t_1}^{(h)}) = \mu_{t_2}(f_{t_2}^{(h)}) = \cdots = \mu_{t_n}(f_{t_n}^{(h)}) = \alpha^{(h)} \ (< \alpha_{PA}).$$

Procedure 6.2

Step 0. By executing Procedure 6.1, let $f^{(0)}(s_i, t_j), f_{s_i}^{(0)}, f_{t_j}^{(0)}$ and $\alpha^{(0)}$ be the resultant flow values in arc (s_i, t_j), from supply s_i, to demand t_j and the satisfaction degree, respectively. Set $h := 1$ and go to Step 1.

Step 1. Find n arcs $(\hat{s}_i, t_j), j = 1, \cdots, n$ for the current flow.

Step 2. After calculating

$$\Delta\mu_{t_j} := \mu_{t_j}(f_{t_j}^{(h-1)}) - \mu_{t_j}(f_{t_j}^{(h-1)} - f^{(h-1)}(\hat{s}_i, t_j)), \ j = 1, \cdots, n,$$

revise the degree of satisfaction $\alpha^{(h-1)}$ as follows:

$$\alpha^{(h)} := \alpha^{(h-1)} - \min_{t_j \in T} \Delta\mu_{t_j}.$$

Step 3. After calculating the revised flow value $f^{(h)}(s_i, t_j)$ (in all arcs (s_i, t_j)) by

$$f^{(h)}(s_i, t_j) := \begin{cases} f^{(h-1)}(s_i, t_j) - f_{t_j}^{(h-1)} + G_{t_j}^{-1}(\alpha^{(h)}) & \text{if } (s_i, t_j) = (\hat{s}_i, t_j), \\ f^{(h-1)}(s_i, t_j) & \text{if } (s_i, t_j) \neq (\hat{s}_i, t_j), \end{cases}$$

the corresponding flow values $f_{s_i}^{(h)}$ and $f_{t_j}^{(h)}$ are calculated as follows:

$$f_{s_i}^{(h)} := \sum_{j=1}^{n} f^{(h)}(s_i, t_j), \ i = 1, \cdots, m,$$
$$f_{t_j}^{(h)} := \sum_{i=1}^{m} f^{(h)}(s_i, t_j), \ j = 1, \cdots, n.$$

Step 4. If $\sum_i \sum_j c_{s_i,t_j} f^{(h)}(s_i,t_j) \leq \overline{C}$, then halt. Otherwise, set $h := h+1$ and return to Step 1.

If this procedure halts at a situation such that $\sum_{i,j} c_{s_i,t_j} f(s_i,t_j) < \overline{C}$, it means that there exists a better solution than the current one since we can improve the degrees of satisfaction at the demand side by increasing f_{t_j}, corresponding to the residual $\overline{C} - \sum_{i,j} c_{s_i,t_j} f(s_i,t_j)$.

Now we assume a situation that $\sum_{i,j} c_{s_i,t_j} f(s_i,t_j) < \overline{C}$ holds after repeating the above iterations k times, i.e., the constraint does not hold until the $(k-1)$-st iteration but it holds after the k-th iteration. Then the values of supplies and demands can be increased from the current values until the total transportation cost becomes equal to the upper limit $\overline{C}$. As shown in the following discussion, it becomes clear that the current solution after the $k-$th iteration (represented by $f_k(s_i,t_j)$, $i=1,\cdots,m, j=1,\cdots,n$) is useless, but the previous (second last) solution obtained in the $(k-1)$-st operation is useful for seeking an optimal solution.

In order to represent a solution of $P10$, we introduce an $m \times n$ solution matrix $(\mathcal{F})$ consisting of $f(s_i,t_j), i=1,\cdots,m, j=1,\cdots,n$. Let $(\mathcal{F}'')$ denote a solution matrix with its satisfation degree α'' such that the total transportation cost is equal to $\overline{C}$, and let $(\mathcal{F}')$ be the $(k-1)$-st solution matrix with its transportation cost C' and satisfaction degree α'. From the viewpoint of basic solution in linear programming, the basic and nonbasic variables corresponding to $(\mathcal{F}'')$ is the same arrangement as those corresponding to the $(k-1)$-st solution $(\mathcal{F}')$ (it should be noted that there exists the degree α'' in the open interval $(\alpha^{(k)}, \alpha^{(k-1)})$). From this fact, the value of demand f''_{t_j} of $(\mathcal{F}'')$ is reduced from $G^{-1}_{t_j}(\alpha')$ (corresponding to $(\mathcal{F}')$) to $G^{-1}_{t_j}(\alpha'')$. That is, we can

find the α'' by solving the following equation:

$$\sum_{j=1}^{n} c_{s_i,t_j} \left\{ G^{-1}_{t_j}(\alpha') - G^{-1}_{t_j}(\alpha'') \right\} = C' - \overline{C}, \tag{6.9}$$

and, based on this, the corresponding flow values $f''(s_i,t_j)$ (also f''_{s_i} and f''_{t_j}) are calculated similarly to Step 3 in Procedure 6.2. Although the total cost computed from these flow values $f(s_i,t_j)''$ is equal to the upper bound $\overline{C}$, it is noted that the solution $(\mathcal{F}'')$ may not be an optimal minimum cost flow. This means that, if there exists flows from plural supply nodes to a demand node in the solution $(\mathcal{F}'')$, we may be able to decrease the total cost by shifting some volume Δd from an arc with a greater cost into that with a smaller, while maintaining the current objective value α''. For example, for two supply nodes s_A and s_B and one demand node t_C, it is assumed that

$$\begin{aligned} &c_{s_A,t_C} < c_{s_B,t_C}, \\ &f''(s_A,t_C) > 0,\ f''(s_B,t_C) > 0, \\ &f''(s_A,t_C) + f''(s_B,t_C) = f''_{t_C}. \end{aligned}$$

In this case, the value of shift Δd from (s_B,t_C) to (s_A,t_C) is determined by the following condition:

$$\mu_{s_A}(f''(s_A,t_C) + \Delta d) \geq \mu_{t_C}(f''_{t_C})\ (=\alpha'').$$

If there exists a value Δd such that this equation holds, flow values are updated as follows:

$$f(s_A,t_C)''' := f(s_A,t_C)'' + \Delta d,\ f(s_B,t_C)''' := f(s_B,t_C)'' - \Delta d.$$

Thus, Procedure 6.2 does not always produce an optimal flow. It is also clear that the total cost C''' resulting from the above improvement is smaller than

$\overline{C}$. Therefore, there may exist feasible solutions whose satisfaction degrees for demand are strictly greater than α''. However, problem $P10$ is solved by developing the discussion in this subsection.

6.2.3 Solution algorithm for $P10$

In order to obtain an optimal flow of $P10$, further investigation of the problem is needed. Now we consider the following parametric linear programming problem $P10'$ such that the total cost is minimized under the constraint that each of the satisfaction degrees at all supply and demand nodes must be greater than or equal to $\alpha \in [0,1]$.

$$P10': \quad \text{Minimize} \quad C = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{s_i,t_j} f(s_i,t_j) \tag{6.10}$$

$$\text{subject to} \quad \sum_{j=1}^{n} f(s_i,t_j) \le F_{s_i}^{-1}(\alpha),\ i=1,\cdots,m, \tag{6.11}$$

$$\sum_{i=1}^{m} f(s_i,t_j) \ge G_{t_j}^{-1}(\alpha),\ j=1,\cdots,n, \tag{6.12}$$

$$\sum_{j=1}^{n} f(s_i,t_j) \ge 0,\ s_i \in S, t_j \in T, \tag{6.13}$$

where the inverse function $F_{s_i}^{-1}(\alpha)$ ($G_{t_j}^{-1}(\alpha)$) gives the value to be shipped from s_i (into t_j) in order that the degree becomes equal to α. Assuming that these functions are given by

$$F_{s_i}^{-1}(\alpha) = p_{s_i} + q_{s_i}\alpha,\ p_{s_i} > 0, q_{s_i} < 0, \tag{6.14}$$

$$G_{t_j}^{-1}(\alpha) = k_{t_j} + l_{t_j}\alpha,\ k_{t_j} > 0, l_{t_j} > 0, \tag{6.15}$$

we consider the problem $D10'$ that is dual to $P10'$:

$$D10': \quad \text{Maximize}\ Z(\alpha) = \sum_{j=1}^{n} w_{t_j}\left(k_{t_j} + l_{t_j}\alpha\right) - \sum_{i=1}^{m} y_{s_i}\left(p_{s_i} + q_{s_i}\alpha\right) \tag{6.16}$$

$$\text{subject to} \quad w_{t_j} - y_{s_i} \le c_{s_i,t_j},\ i=1,\cdots,m, j=1,\cdots,n, \tag{6.17}$$

$$w_{t_j} \ge 0,\ i=1,\cdots,m, \tag{6.18}$$

$$y_{s_i} \ge 0,\ j=1,\cdots,n, \tag{6.19}$$

where w_{t_j} and y_{s_i} are dual variables.

Theorem 6.1 *The objective function of $D12'$ in α is convex.*

Proof. Given $\alpha^{(1)}$ and $\alpha^{(2)}$, let $\alpha^{(\lambda)} = \lambda\alpha^{(1)} + (1-\lambda)\alpha^{(2)}$ where $0 \le \lambda \le 1$, and denote the optimal values for $\alpha^{(\lambda)}, \alpha^{(1)}$ and $\alpha^{(2)}$ by vectors $(w_{t_j}^{(\lambda)}, y_{t_j}^{(\lambda)}), (w_{t_j}^{(1)}, y_{t_j}^{(1)})$ and $(w_{t_j}^{(2)}, y_{t_j}^{(2)})$, respectively. Then

$$\begin{aligned}
Z(\alpha^{(\lambda)}) &= \sum_{j=1}^{n} w_{t_j}^{(\lambda)}\left(k_{t_j} + l_{t_j}\alpha^{(\lambda)}\right) - \sum_{i=1}^{m} y_{s_i}^{(\lambda)}\left(p_{s_i} + q_{s_i}\alpha^{(\lambda)}\right) \\
&= \lambda\left\{\sum_{j=1}^{n} w_{t_j}^{(\lambda)}\left(k_{t_j} + l_{t_j}\alpha^{(1)}\right) - \sum_{i=1}^{m} y_{s_i}^{(\lambda)}\left(p_{s_i} + q_{s_i}\alpha^{(1)}\right)\right\} \\
&\quad +(1-\lambda)\left\{\sum_{j=1}^{n} w_{t_j}^{(\lambda)}\left(k_{t_j} + l_{t_j}\alpha^{(2)}\right) - \sum_{i=1}^{m} y_{s_i}^{(\lambda)}\left(p_{s_i} + q_{s_i}\alpha^{(2)}\right)\right\} \\
&\le \lambda\left\{\sum_{j=1}^{n} w_{t_j}^{(1)}\left(k_{t_j} + l_{t_j}\alpha^{(1)}\right) - \sum_{i=1}^{m} y_{s_i}^{(1)}\left(p_{s_i} + q_{s_i}\alpha^{(1)}\right)\right\} \\
&\quad +(1-\lambda)\left\{\sum_{j=1}^{n} w_{t_j}^{(2)}\left(k_{t_j} + l_{t_j}\alpha^{(2)}\right) - \sum_{i=1}^{m} y_{s_i}^{(2)}\left(p_{s_i} + q_{s_i}\alpha^{(2)}\right)\right\} \\
&= \lambda Z(\alpha^{(1)}) + (1-\lambda)Z(\alpha^{((2)}) \quad \square
\end{aligned}$$

In addition, the objective function $Z(\alpha)$ is piecewise linear and increasing since the first term (of the righthand in (6.16)) $\sum_j w_{t_j}(k_{t_j} + l_{t_j}\alpha)$ increases in α while the second $\sum_i y_{s_i}(p_{s_i} + q_{s_i}\alpha)$ decreases in α, as clear from the definitions of F_{s_i} and G_{t_j}. Thus, the relation between a satisfaction degree α and its minimum transportation cost C, which is indicated by a function $C(\alpha)$, can be represented as shown in Fig.6.3. It is clear from the view point of linear programming that all the solutions corresponding to satisfaction degrees in a same subinterval have the same basic and nonbasic variables. Let α^* be the

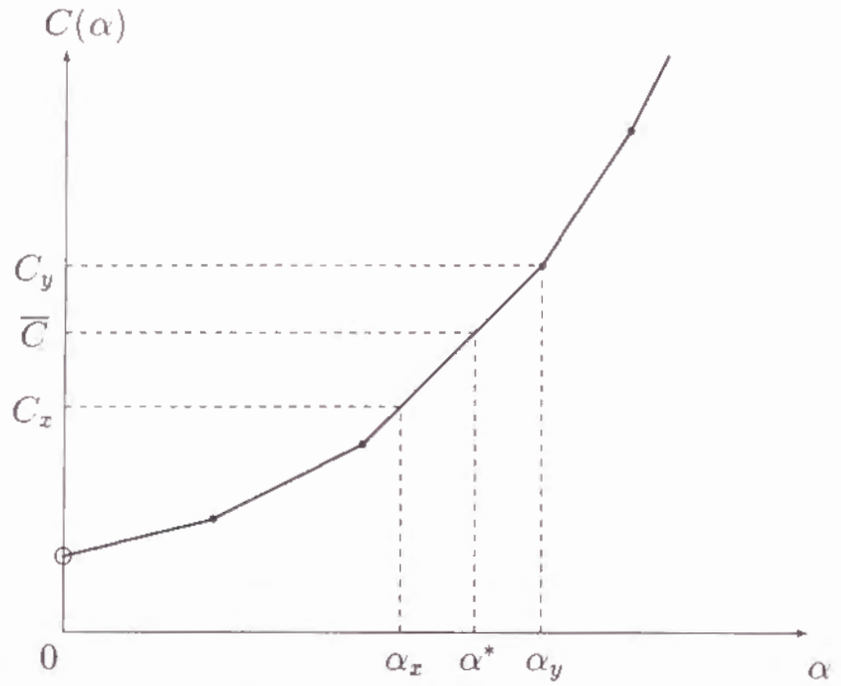

Fig. 6.3 The relation between α and C

optimal satisfaction degree, and α_x and α_y (resp. C_x and C_y) two distinct satisfaction degrees (resp. total costs) satisfying that

$$\alpha_x < \alpha^* < \alpha_y,$$

(see Fig.6.3) and that the corresponding solution matrices $(\mathcal{F}_x)$ and $(\mathcal{F}_y)$ satisfy

$$(\mathcal{F}_x) \doteq (\mathcal{F}_y),$$

where the notation "$\doteq$" denotes that both sides have the same basic and nonbasic variables. Then the same slope of $C(\alpha)$ in the interval $[\alpha_x, \alpha_y]$ is

$$g(\alpha_x, \alpha_y) = \frac{C_y - C_x}{\alpha_y - \alpha_x}. \tag{6.20}$$

Let this interval $[\alpha_x, \alpha_y]$ include α^*. Since the following relation holds,

$$g(\alpha_x, \alpha_y) = g(\alpha_x, \alpha^*) = g(\alpha^*, \alpha_y),$$

we obtain the optimal degree α^* from either of the following equations.

$$\alpha^* = \alpha_x + \frac{\overline{C} - C_x}{g(\alpha_x, \alpha_y)}, \quad \left(\alpha^* = \alpha_y - \frac{C_y - \overline{C}}{g(\alpha_x, \alpha_y)}\right). \tag{6.21}$$

From this α^* and (6.7), we obtain the corresponding supply and demand values $f_{s_i}^{*\prime}$ and $f_{t_j}^{*\prime}$, i.e., $f_{s_i}^{*\prime} := F_{s_i}^{-1}(\alpha^*)$ and $f_{t_j}^{*\prime} := G_{t_j}^{-1}(\alpha^*)$. However, the total sum of $f_{s_i}^{*\prime}$ is not equal to that of $f_{t_j}^{*\prime}$ but $\sum_i f_{s_i}^{*\prime} > \sum_j f_{t_j}^{*\prime}$. Note that this imbalnce occurs whenever we update the flow of a perfect assignment derived from Procedure 6.1 by substituting a satisfaction degree smaller than α_{PA} into (6.7). To prevent this, when we seek a revised minimum cost flow by use of a method for the ordinary transportation problem (e.g., Munkres' method [Mun]), we need to introduce a "dummy node" t_d and m "dummy arcs" (s_i, t_d) whose costs c_{s_i,t_d} are 0. The extended network is represented by $BG' = [S, T'; \mathcal{A}']$. After seeking the α^* from (6.20) and (6.21), we complete the solution procedure by computing a minimum cost flow for the extended network with $f_{s_i}^{*\prime}$ and $f_{t_j}^{*\prime}$. The solution satisfies

$$\mu_{t_1}(f_{t_1}^*) = \cdots = \mu_{t_n}(f_{t_n}^*) = \alpha^* \leq \mu_{s_i}(f_{s_i}^*),\ i = 1, \cdots, m, \tag{6.22}$$

where $f_{s_i}^*$ ($f_{t_j}^*$) are the net flow values to be sent into the demand nodes (sent from the supply nodes) other than t_d, i.e.,

$$\begin{aligned} f_{s_i}^* &:= f_{s_i}^{*\prime} - f(s_i, t_d), \quad i = 1, \cdots, m, \\ f_{t_j}^* &:= f_{t_j}^{*\prime}, \qquad\qquad\quad j = 1, \cdots, n. \end{aligned} \tag{6.23}$$

Therefore, the optimal value of the objective function (6.4) is α^*.

Now we are ready to formally describe an algorithm for $P10$. Note that, in the algorithm, we make use of binary search in order to find the α^* efficiently.

Algorithm 6.1

Step 0. Find a minimum cost flow for the values of supply and demand computed by Procedure 6.1, and let $\alpha_u, (\mathcal{F}_u)$ and C_u be its degree of satisfaction, solution matrix and total transportation cost, respectively. If $C_u \leq \overline{C}$, then terminate (the current flow is optimal). Otherwise, go to Step 1.

Step 1. After setting $\alpha_l := 0, f_{s_i} := b_{s_i}$ and $f_{t_j} := d_{t_j}$, find a minimum cost flow $(\mathcal{F}_l)$ and the corresponding total cost C_l for the revised values f_{s_i} and f_{t_j}. If $C_l > \overline{C}$, terminate (i.e., infeasible). Otherwise, go to Step 2.

Step 2. If $(\mathcal{F}_l) \doteq (\mathcal{F}_u)$, then go to Step 4. Otherwise, update only α_u as $\alpha_u := (\alpha_l + \alpha_u)/2$ and find a minimum cost flow $(\mathcal{F}_u)$ and its total cost C_u for the revised values $f_{s_i} := F_{s_i}^{-1}(\alpha_u)$ and $f_{t_j} := G_{t_j}^{-1}(\alpha_u)$. If $C_u \leq \overline{C}$, go to Step 3 after setting

$$\alpha_l := \alpha_u,\ (\mathcal{F}_l) := (\mathcal{F}_u),\ C_l := C_u,$$

(α_u, $\mathcal{F}_u$ and C_u do not change).

Otherwise, it is divided into two cases, i.e.,

- if $C_u > \overline{C}$ and $(\mathcal{F}_l) \doteq (\mathcal{F}_u)$, then go to Step 4,
- if $C_u > \overline{C}$ and $(\mathcal{F}_l) \not\doteq (\mathcal{F}_u)$, then go to Step 3.

Step 3. After setting $\alpha_m := (\alpha_l + \alpha_u)/2$, find a minimum cost flow $(\mathcal{F}_m)$ and its total cost C_m for the revised values $f_{s_i} := F_{s_i}^{-1}(\alpha_m)$ and $f_{t_j} := G_{t_j}^{-1}(\alpha_m)$. Consider the following four cases, i.e.,

- if $C_m \leq \overline{C}$ and $(\mathcal{F}_m) \doteq (\mathcal{F}_u)$, then go to Step 4 after setting $\alpha_l := \alpha_m$ and $C_l := C_m$,

- if $C_m \leq \overline{C}$ and $(\mathcal{F}_m) \not\doteq (\mathcal{F}_u)$, then return to Step 3 after setting

$$\alpha_l := \alpha_m,\ (\mathcal{F}_l) := (\mathcal{F}_m),\ C_l := C_m,$$

- if $C_m > \overline{C}$ and $(\mathcal{F}_l) \doteq (\mathcal{F}_m)$, then go to Step 4 after setting $\alpha_u := \alpha_m$ and $C_u := C_m$,
- if $C_m > \overline{C}$ and $(\mathcal{F}_l) \not\doteq (\mathcal{F}_m)$, then return to Step 3 after setting

$$\alpha_u := \alpha_m,\ (\mathcal{F}_u) := (\mathcal{F}_m),\ C_u := C_m.$$

Step 4. After calculating $g(\alpha_x, \alpha_y)$ by (6.20), where the variables in the right-hand are set as follows:

$$\alpha_x := \alpha_l,\ \alpha_y := \alpha_u,\ C_x := C_l,\ C_y := C_u,$$

we obtain the optimal degree α^* by (6.21) and find a minimum cost flow $(\mathcal{F}^*)$ for $f_{s_i} := F_{s_i}^{-1}(\alpha^*)$ and $f_{t_j} := G_{t_j}^{-1}(\alpha^*)$. Finally, the optimal values of supply are determined by (6.23). Terminate.

Theorem 6.2 *Algorithm 6.1 solves P10 in $O(ct(m,n) \cdot \log M)$ computational time, where $ct(m,n)$ denotes the time for finding a minimum cost flow and M equals* $\max\{m,n\}$.

Proof. First we describe the validity of Algorithm 6.1 for solving $P10$. Assume $C_u \not\leq \overline{C}$ in Step 1 of the algorithm (otherwise, the current solution is clearly optimal). Then we must decrease the total transportation volume in order to satisfy the upper bound constraint (6.5). At this time, the objective value α of (6.3) should be maintained as great as possible from the view point of the objective function. Accordingly, this modification is equivalent to seeking the

satisfaction degree α^* such that the corresponding objective value C^* of (6.10) in $P10'$ is equal to the upper bound $\overline{C}$ of (6.5) in $P10$. The above disccusion in this subsection assures the validity that we obtain the optimal degree α^* from (6.21). Once the optimal degree α^* is determined, all the f_{s_i} and f_{t_j} are calculated by substituting the α^* into (6.7) and we consequently obtain an optimal minimum cost flow in the extended network $BG' = [S, T'; \mathcal{A}']$ with the values f_{s_i} and f_{t_j} by using a method for the ordinary transportation problem.

Next we show the complexity of Algorithm 6.1. The computational complexity for each step is as follows.

Step 0 : $O(ct(m,n))$ to find a minimum cost flow, since $O(ct(m,n))$ (e.g., Munkres' method: $O(M^3)$ [Mun]) dominates $O(M)$ to calculate m f_{s_i}'s and n f_{t_j}'s.

Step 1 : $O(ct(m,n))$, since this is similar to Step 0.

Step 2 : $O(ct(m,n))$ to find a minimum cost flow, since $O(ct(m,n))$ dominates $O(mn)$ to check relation "$\doteq$" and $O(M)$ to calculate f_{s_i} and f_{t_j}.

Step 3 : $O(ct(m,n))$, similarly to Step 2.

Step 4 : $O(ct(m,n))$, similarly to Step 0.

The "self-loop" of Step 3 is repeated at most $O(\log M)$ times, since it is determined by binary search over at most mn subintervals, as illustrated in Fig.6.3. Therefore, $P10$ can be solved in $O(ct(m,n) \cdot \log M)$ computational time by Algorithm 6.1. □

6.2.4 An example of $P10$

In this subsection, we illustrate Algorithm 6.1 applied to an example of $P10$. Assume that there are three supply nodes s_1, s_2, s_3 and three demand

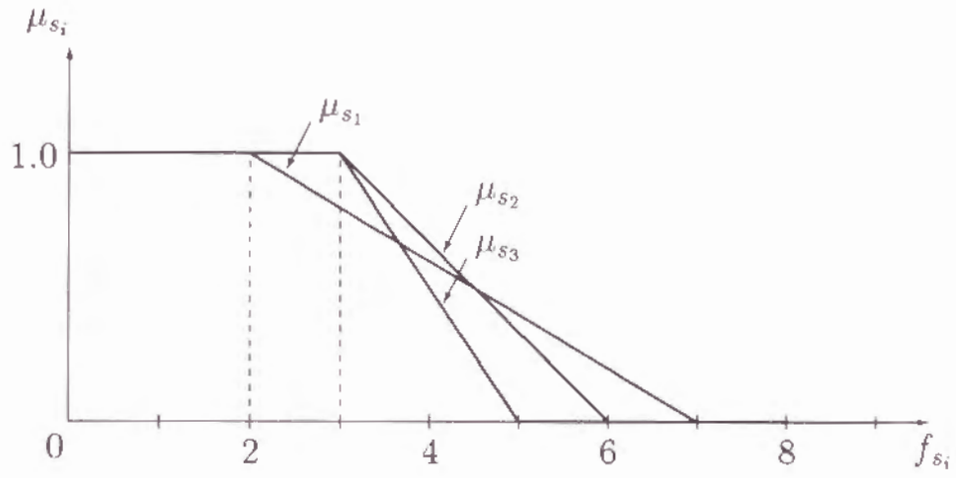

Fig. 6.4 Membership functions $\mu_{s_i}(f_{s_i})$

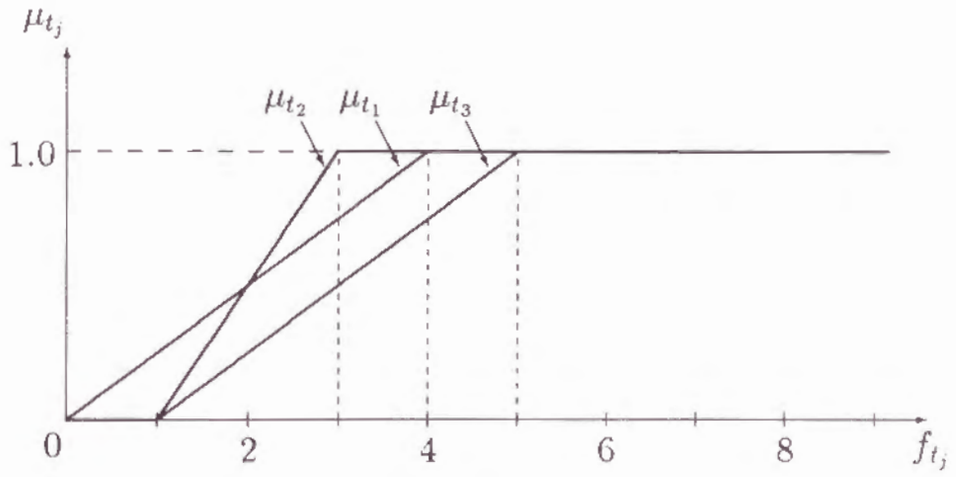

Fig. 6.5 Membership functions $\mu_{t_j}(f_{t_j})$

Tab. 6.1 Transportation cost C_{s_i,t_j}

$s_i \setminus t_j$	1	2	3
1	2	1	5
2	6	3	4
3	4	5	3

nodes t_1, t_2, t_3 whose membership functions are given as follows (see Figs. 6.4 and 6.5):

$$\mu_{s_1}(f_{s_1}) = \begin{cases} 1 & \text{if } f_{s_1} \le 2, \\ -\dfrac{f_{s_1}-7}{5} & \text{if } 2 < f_{s_1} < 7, \\ 0 & \text{if } f_{s_1} \ge 7, \end{cases} \quad \mu_{t_1}(f_{t_1}) = \begin{cases} 0 & \text{if } f_{t_1} \le 0, \\ \dfrac{f_{t_1}}{4} & \text{if } 0 < f_{t_1} < 4, \\ 1 & \text{if } f_{t_1} \ge 4, \end{cases}$$

$$\mu_{s_2}(f_{s_2}) = \begin{cases} 1 & \text{if } f_{s_2} \le 3, \\ -\dfrac{f_{s_2}-6}{3} & \text{if } 3 < f_{s_2} < 6, \\ 0 & \text{if } f_{s_2} \ge 6, \end{cases} \quad \mu_{t_2}(f_{t_2}) = \begin{cases} 0 & \text{if } f_{t_2} \le 1, \\ \dfrac{f_{t_2}-1}{2} & \text{if } 1 < f_{t_2} < 3, \\ 1 & \text{if } f_{t_2} \ge 3, \end{cases}$$

$$\mu_{s_3}(f_{s_3}) = \begin{cases} 1 & \text{if } f_{s_3} \le 3, \\ -\dfrac{f_{s_3}-5}{2} & \text{if } 3 < f_{s_3} < 5, \\ 0 & \text{if } f_{s_3} \ge 5, \end{cases} \quad \mu_{t_3}(f_{t_3}) = \begin{cases} 0 & \text{if } f_{t_3} \le 1, \\ \dfrac{f_{t_3}-1}{4} & \text{if } 1 < f_{t_3} < 5, \\ 1 & \text{if } f_{t_3} \ge 5. \end{cases}$$

The transportation cost c_{s_i,t_j} of each arc (s_i, t_j) is shown in Tab.6.1 (e.g., $c_{s_1,t_1} = 2, c_{s_3,t_2} = 5$) and let the upper limit of total transportation cost be $\overline{C} = 18$.

The 1-st iteration.

Step 0. First, the values of supplies and demands realized a perfect assignment are determined by Procedure 6.1 as follows. From (6.7), we have

$$f_{s_1} = -5\alpha_{PA} + 7,\ f_{s_2} = -3\alpha_{PA} + 6,\ f_{s_3} = -2\alpha_{PA} + 5,$$

$$f_{t_1} = 4\alpha_{PA},\ f_{t_2} = 2\alpha_{PA} + 1,\ f_{t_3} = 4\alpha_{PA} + 1,$$

and $-10\alpha_{PA} + 18 = 10\alpha_{PA} + 2$ holds from (6.8). Hence we obtain $\alpha_{PA} = 4/5 = 0.8$ and the corresponding values are :

$$f_{s_1} := 3,\ f_{s_2} := 18/5,\ f_{s_3} := 17/5,$$

$$f_{t_1} := 16/5,\ f_{t_2} := 13/5,\ f_{t_3} := 21/5.$$

Next, its minimum cost flow is found by using of a method for the ordinary transportation problem, and the resulting solution matrix $(\mathcal{F}_u)$ and total cost C_u in case of $\alpha_u = 4/5 = 0.8$ are as follows:

$$\mathcal{F}_u := \begin{pmatrix} 3 & 0 & 0 \\ 0 & 13/5 & 1 \\ 1/5 & 0 & 16/5 \end{pmatrix},\ C_u := 141/5\ (= 28.2).$$

Since $C_u = 141/5 \not\le \overline{C} = 18$, proceed to Step 1.

Step 1. After setting $\alpha_l := 0, f_{s_i} := b_{s_i}$ and $f_{t_j} := d_{t_j}$, i.e.,

$$f_{s_1} := 7,\ f_{s_2} := 6,\ f_{s_3} := 5,$$

$$f_{t_1} := 0,\ f_{t_2} := 1,\ f_{t_3} := 1,$$

we obtain the following solution matrix $(\mathcal{F}_l)$ and total cost C_l:

$$\mathcal{F}_l := \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},\ C_l := 4.$$

Since $C_l = 4 \not> \overline{C} = 18$, proceed to Step 2.

Step 2. Since $\mathcal{F}_l \ne \mathcal{F}_u$, we find a minimum cost flow for $\alpha_u = (0 + 4/5)/2 = 2/5 = 0.4$:

$$f_{s_1} := 5,\ f_{s_2} := 24/5,\ f_{s_3} := 21/5,$$

$$f_{t_1} := 8/5,\ f_{t_2} := 9/5,\ f_{t_3} := 13/5.$$

$$\mathcal{F}_u := \begin{pmatrix} 8/5 & 9/5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 13/5 \end{pmatrix},\ C_u := 64/5\ (= 12.8).$$

Since $C_u = 64/5 \leq \overline{C} = 18$, proceed to Step 3 after setting

$$\alpha_l := 2/5,\ \mathcal{F}_l := \begin{pmatrix} 8/5 & 9/5 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 13/5 \end{pmatrix},\ C_l := 64/5.$$

Step 3. We obtain a minimum cost flow for $\alpha_m = (2/5 + 4/5)/2 = 3/5 = 0.6$:

$$f_{s_1} := 4,\ f_{s_2} := 21/5,\ f_{s_3} := 19/5,$$

$$f_{t_1} := 12/5,\ f_{t_2} := 11/5,\ f_{t_3} := 17/5.$$

$$\mathcal{F}_m := \begin{pmatrix} 12/5 & 8/5 & 0 \\ 0 & 3/5 & 0 \\ 0 & 0 & 17/5 \end{pmatrix},\ C_m := 92/5\ (= 18.4).$$

Since the current situation satisfies

$$C_m > \overline{C},\ (\mathcal{F}_l) \neq (\mathcal{F}_m),$$

return to Step 3 after setting

$$\alpha_u := 3/5,\ \mathcal{F}_u := \begin{pmatrix} 12/5 & 8/5 & 0 \\ 0 & 3/5 & 0 \\ 0 & 0 & 17/5 \end{pmatrix},\ C_u := 92/5.$$

The 2-nd iteration.

Step 3. A minimum cost flow for $\alpha_m = (2/5 + 3/5)/2 = 1/2 = 0.5$:

$$f_{s_1} := 9/2,\ f_{s_2} := 9/2,\ f_{s_3} := 4,$$

$$f_{t_1} := 2,\ f_{t_2} := 2,\ f_{t_3} := 3.$$

$$\mathcal{F}_m := \begin{pmatrix} 2 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix},\ C_m := 15.$$

Since the current situation satisfies

$$C_m \leq \overline{C},\ (\mathcal{F}_m) \neq (\mathcal{F}_u),$$

return to Step 3 after setting

$$\alpha_l := 1/2,\ \mathcal{F}_l := \begin{pmatrix} 2 & 2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 3 \end{pmatrix},\ C_l := 15.$$

The 3-rd iteration.

Step 3. A minimum cost flow for $\alpha_m = (1/2 + 3/5)/2 = 11/20 = 0.55$:

$$f_{s_1} := 7/4,\ f_{s_2} := 87/20,\ f_{s_3} := 39/10,$$

$$f_{t_1} := 11/5,\ f_{t_2} := 21/10,\ f_{t_3} := 16/5.$$

$$\mathcal{F}_m := \begin{pmatrix} 11/5 & 41/20 & 0 \\ 0 & 1/20 & 0 \\ 0 & 0 & 16/5 \end{pmatrix},\ C_m := 81/5\ (= 16.2).$$

Since the current situation satisfies

$$C_m \leq \overline{C},\ (\mathcal{F}_m) \doteq (\mathcal{F}_u),$$

go to Step 4 after setting

$$\alpha_l := 20/11,\ C_l := 81/5.$$

Step 4. We obtain an optimal satisfaction degree α^* from (6.20) and (6.21), i.e.,

$$g(\alpha_x, \alpha_y) := \left(\frac{92}{5} - \frac{81}{5}\right) \Big/ \left(\frac{3}{5} - \frac{11}{20}\right) = 44,$$

and hence

$$\alpha^* := \frac{3}{5} + \frac{18 - \dfrac{92}{5}}{44} = \frac{13}{22}\ (= 0.5909\cdots).$$

The supply and demand values for α^* are calculated as follows:

$$f_{s_1} := 89/22,\ f_{s_2} := 93/22,\ f_{s_3} := 42/11,$$

$$f_{t_1} := 26/11,\ f_{t_2} := 24/11,\ f_{t_3} := 37/11,$$

and we obtain the following final result:

$$\alpha^* = 13/22,\ \mathcal{F}^* = \begin{pmatrix} 26/11 & 37/22 & 0 \\ 0 & 11/22 & 0 \\ 0 & 0 & 37/11 \end{pmatrix}.$$

Consequently, its total transportation cost is equal to $\overline{C}$, i.e.,

$$\sum_{i=1}^{3}\sum_{j=1}^{3} c_{s_i,t_j} f(s_i,t_j) = \frac{26\cdot 2}{11} + \left(\frac{37\cdot 1}{22} + \frac{11\cdot 3}{22}\right) + \frac{37\cdot 3}{11} = 18.$$

6.3 Fuzzy transportation problem with integral flow

6.3.1 Problem formulation

In this subsection, we consider a variant of the fuzzy transportation problem, in which it is assumed that all supply and demand values are integers and that the values of commodities to be transported are integers.

To formulate the problem, we use the same notations as in the previous section, except that the following integer constraint (6.27) is imposed in place of the cost constraint (6.5) of $P10$, i.e.,

$$P11: \quad \text{Maximize} \quad \min\left\{\mu_{s_i}(f_{s_i}), \mu_{t_j}(f_{t_j}) \mid s_i \in S, t_j \in T\right\}, \tag{6.24}$$

$$\text{subject to} \quad \sum_{t_j\in T} f(s_i,t_j) = f_{s_i},\quad i = 1,\cdots,m,$$
$$\sum_{s_i\in S} f(s_i,t_j) = f_{t_j},\quad j = 1,\cdots,n, \tag{6.25}$$

$$f(s_i,t_j) \geq 0,\ s_i \in S, t_j \in T, \tag{6.26}$$

$$f_{s_i}, f_{t_j}: \ \text{non-negative integers},\quad i = 1,\cdots,m,\ j = 1,\cdots,n. \tag{6.27}$$

Note that these f_{s_i} and f_{t_j} are not constants but variables to be determined. In the sense of the ordinary transportaion problem, it may also be desired to introduce the constraint (6.5) saying that the total transportation cost is not larger than a given constant. However, since taking into account both constraints (6.5) and (6.27) makes the problem intractable, our problem does not involve the concept of cost. As a result, there may exist more than one optimal solution with the same objective value. To make the optimal solution unique, we shall additionally find the solution such that its transportation cost is the minimum among all minimum costs of the flow patterns which realize the same objective value.

6.3.2 Solution procedure for $P11$

In order to solve $P11$, we first make use of Procedure 6.1. If all values of supplies and demands obtained by Procedure 6.1 are integers, it is clearly optimal in the sense of $P11$ and the corresponding flow pattern is immediately found by any solution method for the ordinary transportaion problem. Otherwise, we must find the optimal integral values f_{s_i} and f_{t_j} such that the minimum degree of all the corresponding membership values is greatest among all feasible solutions, and then obtain the corresponding flow that minimizes the total transportation cost by applying any solution method for the ordinary transportation problem to the bipartite network with the optimal f_{s_i} and f_{t_j}.

Now we consider how to compute the integral f_{s_i} and f_{t_j} after Procedure 6.1. We first update all the current values of f_{s_i} and f_{t_j} as follows:

$$\tilde{f}_{s_i} := \lfloor f_{s_i} \rfloor,\quad i = 1,\cdots,m,$$
$$\tilde{f}_{t_j} := \lceil f_{t_j} \rceil,\quad j = 1,\cdots,n, \tag{6.28}$$

where $\lfloor f_{s_i} \rfloor$ is the greatest integer not greater than f_{s_i} and $\lceil f_{t_j} \rceil$ is the smallest

integer not smaller than f_{t_j}. Then, all membership values for the revised values $\tilde{f}_{s_i}$ and $\tilde{f}_{t_j}$ are not smaller than those for the previous values (i.e., $\mu_{s_i}(\tilde{f}_{s_i}) \geq \mu_{s_i}(f_{s_i})$ and $\mu_{t_j}(\tilde{f}_{t_j}) \geq \mu_{t_j}(f_{t_j})$ for all i and j). However, it causes an imbalance between the total value of supplies and that of demands (i.e., $\sum_{s_i \in S} \tilde{f}_{s_i} < \sum_{t_j \in T} \tilde{f}_{t_j}$), and hence it is infeasible since there exists the sink node $t_{j'}$ such that $\sum_{s_i \in S} f(s_i, t_{j'}) < \tilde{f}_{t_{j'}}$. To remove the imbalance, we execute the following two possible operations:

(A) to increase $\tilde{f}_{s_i}$, (B) to decrease $\tilde{f}_{t_j}$

by integral values. Since both of them decrease their satisfaction degrees, we have to select a set of the values such that the objective value of (6.24) is greatest.

Let k be a value of the imbalance, i.e.,

$$k = \sum_{j=1}^{n} \lceil f_{t_j} \rceil - \sum_{i=1}^{m} \lfloor f_{s_i} \rfloor, \tag{6.29}$$

where k is a positive integer. In order to examine the resulting membership values by the above operations (A) and (B), we introduce useful notations for executing the operations. Let $P_{i,h}$ ("P" of Plant) and $W_{j,l}$ ("W" of Warehose) be the membership values of $\mu_{s_i}(h)$ and $\mu_{t_j}(l)$, where h and l are integers such that $\tilde{f}_{s_i} + 1 \leq h \leq \tilde{f}_{s_i} + k$ and $\tilde{f}_{t_j} - 1 \geq l \geq \tilde{f}_{t_j} - k$, respectively. That is,

$$\begin{aligned} P_{i,h} &:= \mu_{s_i}(\tilde{f}_{s_i} + v), \quad i = 1, \cdots, m,\ v = 1, \cdots, k, \\ W_{j,l} &:= \mu_{t_j}(\tilde{f}_{t_j} - v), \quad j = 1, \cdots, n,\ v = 1, \cdots, k. \end{aligned} \tag{6.30}$$

After obtaining these km $P_{i,h}$'s and kn $W_{j,l}$'s, we sort them by using "merge sort" in the non-increasing order. The resulting sequence is represented by

$$\mu^{(1)} \geq \mu^{(2)} \geq \cdots \geq \mu^{(km+kn)}, \tag{6.31}$$

for the convenience of discussion. Note that each of these elements is either $P_{i,h}$ or $W_{j,l}$. It is clear that only the k largest elements $\mu^{(1)} \geq \cdots \geq \mu^{(k)}$ among

them are necessary in order to seek a set of the optimal values of supplies and demands. That is, $P_{i',h}$ and $W_{j',l}$ (for $s_{i'} \in S$ and $t_{j'} \in T$) in these k elements indicate that supplies $\tilde{f}_{s_{i'}}$ should be increased by $h - \tilde{f}_{s_{i'}}$ and demands $\tilde{f}_{t_{j'}}$ should be decreased by $\tilde{f}_{t_{j'}} - l$, respectively. If there exists more than one element for some supply node in (6.31), the last operation executed in the order of $\mu^{(1)} \rightarrow \mu^{(2)} \rightarrow \cdots$ of (6.31) is available.

For example, in case $k = 3$ and the sequence of (6.31) is

$$\mu^{(1)} = P_{2,6} \geq \mu^{(2)} = W_{1,5} \geq \mu^{(3)} = P_{2,7},$$

we obtain the following result:

$$f_{s_2} := 7,\ f_{t_1} := 5.$$

Note that set $f_{s_i} := \tilde{f}_{s_i}$ and $f_{t_j} := \tilde{f}_{t_j}$ for $i \neq 2$ and $j \neq 1$. In case there exists more than one element for some demand node in (6.31), we can obtain the result by executing the operation similarly to the ahove case for supply node. Thus the values of f_{s_i} and f_{t_j} to be determined are obtained. Then we can seek the flow pattern that minimizes the total transportation cost.

However, since there may exist "ties" for the $k-$th element in (6.31), i.e.,

$$\mu^{(1)} \geq \cdots \geq \mu^{(k)} = \mu^{(k+1)} = \cdots = \mu^{(k+q)}, \tag{6.32}$$

where q denotes an integer that indicates the number of ties. In this case, if we execute all $(k+q)$ operations then the resulting values f'_{s_i} and f'_{t_j} after executing these $(k+q)$ operations may not remove the imbalance, but $\sum_i f'_{s_i} \geq \sum_j f'_{t_j}$ holds from the definition of membership functions (6.1) and (6.2). This inequality implies that we can use a method for the ordinary transportation problem by introducing a "dummy node" and m "dummy arcs" (recall the extended network BG' in the previous section). Therefore, in case there exist

q ties for the k-th element in (6.31), we obtain the optimal values of f_{s_i} and f_{t_j} from the resulting net flow values, which are calculated by excluding the flow values in m dummy arcs and satisfy the constraints (6.25) through (6.27). Now we are ready to describe an algorithm for solving $P11$; its validity will be shown in the proof of Theorem 6.3.

Algorithm 6.2

Step 0. Compute the supply and demand values that realize a perfect assignment by use of Procedure 6.1. If all the values are integers, the current values are optimal. The corresponding flow and its minimum total cost are computed by applying a suitable method for the ordinary transportation problem, and terminate. Otherwise, go to Step 1.

Step 1. Compute $\tilde{f}_{s_i}$ and $\tilde{f}_{t_j}$ of (6.28), and tbe value k of imbalance is calculated by (6.29).

Step 2. Compute km $P_{i,h}$'s and kn $W_{j,l}$'s by (6.30) and then sort them in the non-increasing order. Select the k largest elements $P_{i,h}$ and $W_{j,l}$, and store the values of supplies and demands corresponding to them in a set ST, i.e.,

$$ST := (f'_{s_1}, \cdots, f'_{s_m}; f'_{t_1}, \cdots, f'_{t_n}),$$

where f'_{s_i} and f'_{t_j} denote the supply and demand values after executing these k operations. At this time, in case there are q ties for the k-th element, also the $(k+1)$-st through $(k+q)$-th operations are additionally executed for the setting of ST.

Step 3. Compute the minimum cost flow by applying a method for the ordinary transportation problem to the extended network BG' with respect to ST of Step 2. We obtain the unique solution of $f^*_{s_i}$ and $f^*_{t_j}$ from the resulting minimum cost flow, i.e.,

$$f^*_{s_i} := \sum_{j=1}^{n} f'(s_i, t_j),\ i = 1, \cdots, m,$$

$$f^*_{t_j} := \sum_{i=1}^{m} f'(s_i, t_j),\ j = 1, \cdots, n,$$

where $f'(s_i, t_j)$ denotes the flow in arc (s_i, t_j) of the resulting flow. Terminate.

Theorem 6.3 *Algorithm 6.2 solves $P11$ in $O(\max(ct(m,n), M^2 \log M))$ computational time, where $ct(m,n)$ denotes the time for finding a minimum cost flow and $M = \max\{m,n\}$.*

Proof. First we describe the validity of Algorithm 6.2. In this proof, we use the notations $f^{**}_{s_i}$ and $f^{**}_{t_j}$ as the values of supply and demand computed in Step 0 for the convenience of discussion. If all the values are integers, the current values are clearly optimal. In this case, we immediately obtain the corresponding minimum cost flow by applying a method for the ordinary transportation problem to the network with the integral values. On the other hand, if there exist supply values $f^{**}_{s_i}$ (demand values $f^{**}_{t_j}$) such that $f^{**}_{s_i} \neq \lfloor f^{**}_{s_i} \rfloor$ ($f^{**}_{t_j} \neq \lceil f^{**}_{t_j} \rceil$), we have to update them to integer values. After updating them by $\tilde{f}_{s_i} := \lfloor f^{**}_{s_i} \rfloor$ and $\tilde{f}_{t_j} := \lceil f^{**}_{t_j} \rceil$, the resulting values are all integers, and membership values $\mu_{s_i}(\tilde{f}_{s_i})$ and $\mu_{t_j}(\tilde{f}_{t_j})$ are not smaller than the previous values $\mu_{s_i}(f^{**}_{s_i})$ and $\mu_{t_j}(f^{**}_{t_j})$, respectively. However, it causes the imbalance $k\ (> 0)$ of (6.29). The imbalance $\sum_{s_i \in S} \tilde{f}_{s_i} < \sum_{t_j \in T} \tilde{f}_{t_j}$ implies that there exists sink node $t_{j'}$ such that the net flow value $\sum_{s_i \in S} f(s_i, t_{j'})$ sent into $t_{j'}$ is smaller than the revised value $\tilde{f}_{t_{j'}}$. That is, the revised values $\tilde{f}_{s_i}$ and $\tilde{f}_{t_j}$

do not satisfy the constraints of (6.25). In order to remove the imbalance, we should execute the operations based on $P_{i,h}$ and $W_{j,l}$ of (6.30), namely (A) to increase $\tilde{f}_{s_i}$ and/or (B) to decrease $\tilde{f}_{t_j}$ by integral values. Then, Step 2 selects the k largest elements among the $k(m+n)$ elements $P_{i,h}$ and $W_{j,l}$, which are sorted as (6.31), and execute the operations (A) and (B) corresponding to these k elements.

Now we assume that there exist q ties for the k-th element in (6.31), and let α be the membership value corresponding to the k-th element $\mu^{(k)}$, i.e.,

$$\mu^{(1)} \geq \cdots \geq \mu^{(k)} = \mu^{(k+1)} = \cdots = \mu^{(k+q)} \triangleq \alpha.$$

In this case, we show that an optimal membership value is α, which maximizes the minimum among all satisfaction degrees for the supplies sent out from the source nodes and demands sent into the sink nodes. Let k' and k'' be the values such that

$$k' < k,$$
$$\mu^{(k')} > \mu^{(k)},$$

and

$$k'' > k,$$
$$\mu^{(k'')} < \mu^{(k)},$$

respectively. If k' operations of $\mu^{(1)}$ through $\mu^{(k')}$ are executed, the imbalance remains by $k - k' > 0$. On the other hand, k'' operations of $\mu^{(1)}$ through $\mu^{(k'')}$ are clearly excessive, since the resulting objective value after executing these k'' operations becomes $\mu^{(k'')}$ $(< \mu^{(k)} = \alpha)$. Hence, the optimal objective value is α even if there are q ties for the k-th element in (6.31).

To make the optimal solution of f_{s_i} and f_{t_j} unique, we choose the solution such that its transportation cost is the minimum among all minimum costs of the flow patterns which realize the satisfaction degree α. These flow patterns and their minimum costs are obtained by applying a method for the ordinary transportation problem to the network with the values of supply and demand after executing arbitrary k operations among $\mu^{(1)}$ through $\mu^{(k+q)}$. In this manner, we must find a minimum cost flow ${}_{k+q}C_k = k(k-1)\cdots\cdots(q+1)/(1\cdot 2\cdots\cdots k)$ times and examine their minimum costs one by one. However, this manner is not so efficient and the unique optimal solution can be determined by applying a method for the ordinary transportation problem just once after executing all the operations corresponding to $\mu^{(1)}$ through $\mu^{(k+q)}$.

To prove this, we use the following notations. Let $M^{(k+q)}$ be a set of all the elements $\{\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(k+q)}\}$, and a set of k elements which are arbitrarily selected among $M^{(k+q)}$ is denoted by $M^{(k)}$. Moreover, let $C^{(k)}$ and $C^{(k+q)}$ be the value of minimum cost flow after executing k operations in $M^{(k)}$ and that after executing $k+q$ operations in $M^{(k+q)}$, respectively. By executing not only k operations in $M^{(k)}$ but also some $W_{j,l}$ in the set of $M^{(k+q)} - M^{(k)}$, new imbalance $\sum_{s_i \in S} f'_{s_i} > \sum_{t_j \in T} f'_{t_j}$ arises, where f'_{s_i} and f'_{t_j} denote the modified values of supply and demand after executing these operations. However, all the excessive flow concerning this imbalance, i.e., $\sum_{s_i \in S} f'_{s_i} - \sum_{t_j \in T} f'_{t_j} > 0$, is sent into dummy node t_d by applying a method for the ordinary transportation problem. As a result, the minimum total cost for the modified values f'_{s_i} and f'_{t_j} is clearly smaller than $C^{(k)}$. Note that the resulting objective value becomes α. While, in case we execute not only k operations in $M^{(k)}$ but also some $P_{i,h}$ in the set of $M^{(k+q)} - M^{(k)}$, the minimum total cost for the modified values is smaller than or equal to $C^{(k)}$ for a reason similar to the above case. Note that if the increase of f_{s_i} arisen by executing $P_{i,h}$ in $M^{(k+q)} - M^{(k)}$ does not decrease

the toal cost $C^{(k)}$, the excessive flow concerning these $P_{i,h}$ is sent into dummy node t_d, and consequently the minimum total cost is equal to $C^{(k)}$. Therefore, if there exist q ties for the $k-$th element in (6.31), we obtain the unique solution $f^*_{s_i}$ and $f^*_{t_j}$ by applying a method for the ordinary transportation problem after all the $k+q$ operations corresponding to $\mu^{(1)}$ through $\mu^{(k+q)}$ are executed. That is, the unique solution $f^*_{s_i}$ and $f^*_{t_j}$ is obtained as follows.

$$f^*_{s_i} := \sum_{j=1}^{n} f'(s_i, t_j),\ i = 1, \cdots, m,$$
$$f^*_{t_j} := \sum_{i=1}^{m} f'(s_i, t_j),\ j = 1, \cdots, n,$$

where $f'(s_i, t_j)$ denotes the flow in arc (s_i, t_j) computed by applying a method for the ordinary transportation problem to the network with the supply and demand values modified by executing all the $k + q$ operations.

Next we analyze the complexity of Algorithm 6.2. The required time for each step is as follows.

Step 0 : $O(M)$.

Step 1 : $O(M)$.

Step 2 : $O(M^2 \log M)$ to sort km $P_{i,h}$'s and kn $W_{j,l}$'s, since $k \leq m + n - 2$ and hence at most $(m + n)(m + n - 2)$, i.e., $O(M^2)$ elemnts are sorted, and $O(M^2)$ to set all candidates.

Step 3 : $O(ct(m, n))$ to find a minimum cost flow.

Therefore, problem $P11$ can be solved in $O(\max(ct(m, n), M^2 \log M))$ computational time by Algorithm 6.2. □

6.3.3 An example of $P11$

We illustrate Algorithm 6.2 applied to an example of $P11$. Assume that there are two supply nodes s_1, s_2 and three demand nodes t_1, t_2, t_3 and the transportation costs of arcs (s_i, t_j) are shown in Tab.6.2. Each membership function is given as follows (see Figs. 6.6 and 6.7):

$$\mu_{s_1}(f_{s_1}) = \begin{cases} 1 & \text{if } f_{s_1} \leq 2, \\ -\dfrac{f_{s_1} - 7}{5} & \text{if } 2 < f_{s_1} < 7, \\ 0 & \text{if } f_{s_1} \geq 7, \end{cases}$$
$$\mu_{s_2}(f_{s_2}) = \begin{cases} 1 & \text{if } f_{s_2} \leq 3, \\ -\dfrac{f_{s_2} - 6}{3} & \text{if } 3 < f_{s_2} < 6, \\ 0 & \text{if } f_{s_2} \geq 6, \end{cases}$$
$$\mu_{t_1}(f_{t_1}) = \begin{cases} 0 & \text{if } f_{t_1} = 0, \\ \dfrac{f_{t_1}}{4} & \text{if } 0 < f_{t_1} < 4, \\ 1 & \text{if } f_{t_1} \geq 4, \end{cases}$$
$$\mu_{t_2}(f_{t_2}) = \begin{cases} 0 & \text{if } f_{t_2} = 0, \\ \dfrac{f_{t_2}}{5} & \text{if } 0 < f_{t_2} < 5, \\ 1 & \text{if } f_{t_2} \geq 5, \end{cases}$$
$$\mu_{t_3}(f_{t_3}) = \begin{cases} 0 & \text{if } f_{t_3} \leq 1, \\ \dfrac{f_{t_3} - 1}{5} & \text{if } 1 < f_{t_3} < 6, \\ 1 & \text{if } f_{t_3} \geq 6. \end{cases}$$

Tab. 6.2 Transportation cost C_{s_i,t_j}

$s_i \setminus t_j$	1	2	3
1	2	1	5
2	6	4	3

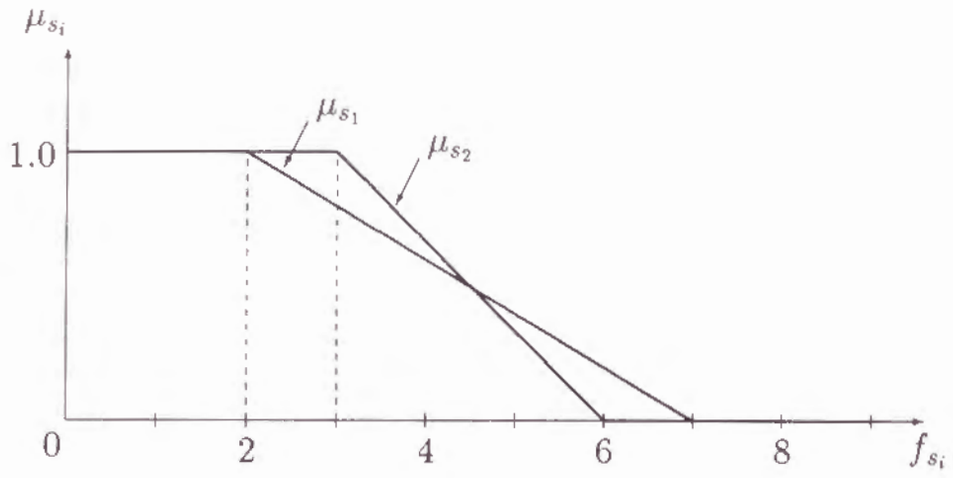

Fig. 6.6 Membership functions $\mu_{s_i}(f_{s_i})$

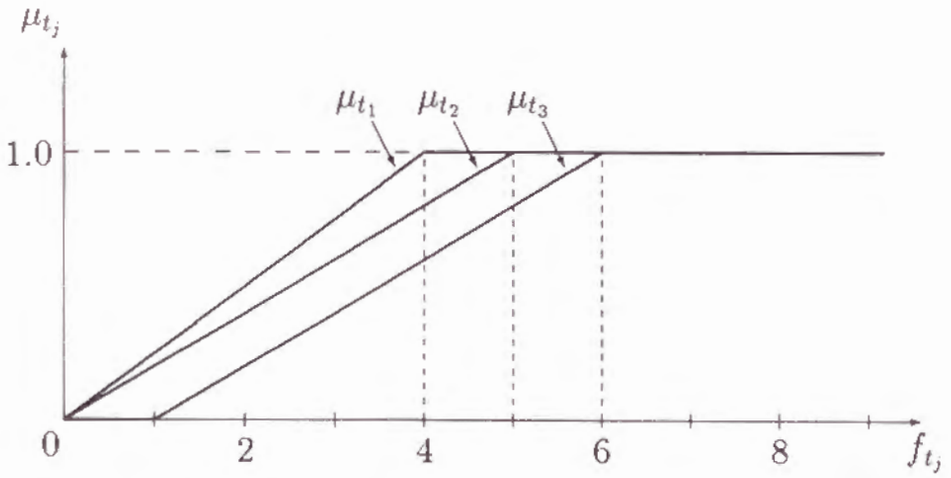

Fig. 6.7 Membership functions $\mu_{t_j}(f_{t_j})$

The 1-st iteration.

Step 0. By Procedure 6.1, we obtain the following result.

$$\alpha_{PA} = 6/11\ (= 0.5454\cdots),$$
$$f_{s_1} = 47/11\ (= 4.2727\cdots),$$
$$f_{s_2} = 48/11\ (= 4.3636\cdots),$$
$$f_{t_1} = 24/11\ (= 2.1818\cdots),$$
$$f_{t_2} = 30/11\ (= 2.7272\cdots),$$
$$f_{t_3} = 41/11\ (= 3.7272\cdots).$$

Since none of these values are integers, proceed to Step 1.

Step 1. By (6.28), we have

$$(\check{f}_{s_1}, \check{f}_{s_2}; \check{f}_{t_1}, \check{f}_{t_2}, \check{f}_{t_3}) := (4, 4; 3, 3, 4),$$

and the value k of imbalance (6.29) is as follows:

$$k := (3 + 3 + 4) - (4 + 4) = 2.$$

Step 2. By (6.30), $P_{i,h}$ and $W_{j,l}$ are calculated as follows:

$$P_{1,5} := 2/5,\ P_{1,6} := 1/5,\ P_{2,5} := 1/3,\ P_{2,6} := 0,$$
$$W_{1,2} := 1/2,\ W_{1,1} := 1/4,\ W_{2,2} := 2/5,\ W_{2,1} := 1/5,$$
$$W_{3,3} := 2/5,\ W_{3,2} := 1/5,$$

and these values are sorted in the non-increasing order, i.e.,

$$W_{1,2} = 1/2 \geq W_{3,3} = 2/5 \geq W_{2,2} = 2/5$$
$$\geq P_{1,5} = 2/5 \geq P_{2,5} = 1/3 \geq \cdots \geq P_{2,6} = 0.$$

Since there exist ties for the 2-nd elements, namely $W_{3,3} = W_{2,2} = P_{1,5} = 2/5$ (i.e., an optimal value of the objective function is 2/5), we obtain the following result:

$$ST = (f'_{s_1}, f'_{s_2}; f'_{t_1}, f'_{t_2}, f'_{t_3}) := (5, 4; 2, 2, 3).$$

Proceed to Step 3.

Step 3. The minimum cost flow (solution matrix) $\mathcal{F}^*$ and its total cost C^* with respect to ST are computed by use of a method for the ordinary transportation problem, as shown below:

$$\mathcal{F}^* := \begin{pmatrix} 2 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \ C^* := 15.$$

Therefore, the unique solution of $f^*_{s_i}$ and $f^*_{t_j}$ is

$$(f^*_{s_1}, f^*_{s_2};\ f^*_{t_1}, f^*_{t_2}, f^*_{t_3}) = (4, 3;\ 2, 2, 3),$$

and the objective value is 2/5 since we have

$$\mu_{s_1}(4) = 3/5, \mu_{s_2}(3) = 1, \mu_{t_1}(2) = 1/2, \mu_{t_2}(2) = 2/5, \mu_{t_3}(3) = 2/5.$$

Chapter 7

CONCLUSION

In this dissertation, we have discussed the fuzzifications of some combinatorial optimization problems. It is important to see how the problems in fuzzy environments are formulated, especially how the optimality of such problems is defined. For these purposes, we have introduced new fuzzy criteria instead of the ordinary rigid ones. After formulating fuzzy combinatorial optimization problems in this manner, efficient algorithms have been proposed. Here we summarize the contribution of this dissertation, and then discuss future research topics.

The contents of this dissertation are divided into two parts. PART I consists of Chapters 2 through 4 and discusses some types of fuzzy scheduling problems. In Chapter 2, we have introduced three fuzzy factors into scheduling models, i.e., fuzzy due-dates, fuzzy processing times and fuzzy precedence constraints, which are based on the concepts of degree of satisfaction, fuzzy number and fuzzy relation, respectively. The idea of fuzzy due-dates may be interpreted as the expression of uncertain or ambiguous due-dates similar to fuzzy number. That is, fuzzy due-dates can be interpreted as the values approximately equal or smaller than the values of dead-lines in the ordinary sense.

In Chapter 3, we have formulated fuzzy scheduling problems on a single

machine, and proposed efficient algorithms to solve them. For the first fuzzy factor, the problem $n \mid 1 \mid L_{\max}$ with fuzzy due-dates has been formulated as the problem that maximizes the minimum value of satisfaction degrees for completion times. An optimal solution of this problem is obtained from the well-known EDD rule after the max-min value of the objective function is determined by use of binary search technique over the interval $[0, 1]$. That is, once the objective value is fixed, the problem is reduced to the ordinary problem $n \mid 1 \mid L_{\max}$. This has been generalized to the problems with weighted fuzzy due-dates by introducing the weights that represent the importance of the corresponding jobs. The generalized problems can also be solved in a manner similar to the previous problem.

For the second fuzzy factor, i.e., fuzzy processing times, we have introduced the idea of agreement index of fuzzy set theory as a measure of its optimality criterion, and formulated the problem $n \mid 1 \mid L_{\max}$ with fuzzy processing times as the problem that maximizes the minimum value among all agreement indices. This indicates the evaluation of lateness. In this formulation, it is assumed that all fuzzy processing times are "quasi-similar" triangular fuzzy numbers, and an optimal schedule is obtained from the EDD-order. The problem without the assumption of "quasi-similarlity" is left as a topic of future research.

As the problem with the third fuzzy factor in scheduling models, the problem $n \mid 1 \mid L_{\max}$ with fuzzy precedence constraints has been investigated. While the original problem (namely $n \mid 1 \mid L_{\max}$ with precedence relations) has to satisfy the constraint of precedence relations, its fuzzy version tolerates violation of precedence constraints within the satisfaction level indicated by the associated job-pairs. Generally speaking, if all precedence constraints are excluded from the original problem, the optimal $L_{\max}$ value will become

smaller than that of constrained case. Accordingly, we have formulated problem $n \mid 1 \mid L_{\max}$ with fuzzy precedence constraints as the bi-criteria problem, i.e., $L_{\max}$ to be minimized and the minimum satisfaction level with respect to fuzzy precedece relation to be maximized. The problem is solved by using the concept of non-dominated schedule. We expect that this version of fuzzy precedence can also be extended to many other problems, e.g., the problems with tree type precedence relation of multi identical machines, mean flow time minimization and so on.

Chapter 4 has dealt with two fuzzy scheduling problems on multi-machines. First we have discussed the problem on identical machines $n \mid m \mid I \mid L_{\max}$ with fuzzy general due-dates in the sense that each job has its own fuzzy due-date for each machine. The objective is to maximize the minimum satisfaction degrees for completion times. We have shown that the problem can be solved by using the reduced network and the idea of modified due-dates, which are then combined with binary search technique. Once the optimal value is determined, the corresponding solution is obtained from the well-known algorithm of Gonzalez and Sahni.

Next we have investigated the fuzzification of the two machine open shop problem $n \mid 2 \mid O \mid L_{\max}$. This problem is first generalized by allowing each machine speed to be controllable, and then further generalized by introducing fuzzy due-dates. The aim is to determine an optimal speed of each machine and to obtain an optimal schedule with respect to the objective function that consists of the sum of the minimum satisfaction degree among all jobs and the cost of machine speeds. We have clarified that an optimal solution of this problem is obtained by decomposing it into subproblems and then finding the best one among all the optimal solutions of these subproblems.

Some algorithms proposed in this part are not efficient enough, and their

refinements are needed. For fuzzy scheduling, we think that many promising and interesting regions remain to be investigated.

PART II of this dissertation consists of Chapters 5 and 6, and deals with fuzzy network problems. The fuzzy factors proposed here are based on the concept of degree of satisfaction. However, fuzzy network problems with other factors may also be worth investigating. For example, introduction of fuzzy relation to network problems, such as the assignment problem, may be promising.

In Chapter 5, first we have investigated the fuzzy sharing problem, in which all sinks have fuzzy weights based on the concept of fuzzy objective function. That is, the objective is to maximize the minimum among all degrees of satisfaction for the total flow sent into sink nodes. We have shown by fully using the well-known max-flow min-cut theorem that the proposed algorithm provides an optimal solution, which is optimal also in the sense of min-max sharing. The fuzzification of the sharing problem is particularly meaningful in case some sink nodes are satisfied with a value less than the exact optimal one and the surplus may be sent into other sink nodes which are not satisfied with the exact optimal values.

Next we have investigated a generalized fuzzy sharing problem under the additional constraint of "block-unit". We have shown that the generalized problem can be solved in a manner similar to the fuzzy sharing problem, but the optimal flow for the max-min sharing is in general different from that for the min-max sharing.

In Chapter 6, we have discussed the fuzzy transportation problem whose objective is to maximize the minimum among all degrees of satisfaction for the supplies, which are sent out from the source nodes, and the demands, which are sent into the sink nodes, under the upper limit constraint of the

total transportation cost. We have proposed an algorithm that traces basic solutions in parametric linear programming. When we interpret the fuzzy supplies as the values approximately smaller than or equal to the possible supplies, and the fuzzy demands as the values approximately greater than or equal to the required demands, respectively, the fuzzy transportation problem can be formulated and solved even though the total amount of demands is larger than that of supplies. In these situations, fuzzification seems to make sense.

Further, we have dealt with the problem on which integral flow constraint is imposed in place of the total cost constraint of the fuzzy transportation problem. The problem with integral flow is solved by introducing the useful notion to narrow the candidates for the optimal solution.

In the real situations, making a decision involves more or less human cognitive process at least in its final step, and this implies that there probably exists the case which a decision maker can not directly adopt the plans obtained from the solutions of ordinary (non-fuzzy) optimization problems. We think that it is therefore important to investigate fuzzy combinatorial optimization. The author hopes that the work contained in this dissertation will contribute to the further development of the field of fuzzy combinatorial optimization and help more flexible and useful decision making in the real problems.

References

[Aho] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The design and analysis of computer algorithm*, (Adddison-Wesley, Reading, Mass., 1974).

[Bel] R.E. Bellmann and L.A. Zadeh, "Decision making in a fuzzy environment", *Management Science*, Vol.17 (1970) 141-164.

[Bla] J. Blazewicz, "Selected topics in scheduling theory", *Annals of Discrete Mathematics*, 31 (1987) 1-60.

[Bro] J. R. Brown, "The sharing problem", *Operations Reseach*, Vol. 27, No. 2 (1979) 324-340.

[Cha] S.Chanas and W.Kołodziejczyk, "Maximum flow in a network with fuzzy arc capacities", *Fuzzy Sets and Systems*, 8 (1982) 165-173.

[Deg] M. H. Degroot, *Optimal statistical decisions*, (McGraw-Hill, New York, 1970).

[Din] E.A.Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation", *Sovieet Math. Dokl*, 11 (1970) 1277-1280.

[For] L.R.Ford,Jr. and D.R.Fulkerson, *Flows in network*, (Princeton University Press, 1962).

[Gon] T.Gonzalez and S.Shani, "Open shop scheduling to minimize finish time", *Journal of ACM.*, 23 (1976) 665-679

[Han] E.L. Hannan, "Linear programming with multiple fuzzy goals", *Fuzzy Sets and Systems*, Vol.6, (1981) 235-248.

[Inu] M. Inuiguchi, "Stochastic programming problems versus fuzzy mathematical programming problems", *J. of Japan Society for Fuzzy Theory and Systems*, Vol.4, No.1 (1992) 21-30. (in Japanese)

[Isb] H. Ishibuchi, "Fuzzy regression analysis", *J. of Japan Society for Fuzzy Theory and Systems*, Vol.4, No.1 (1992) 52-60. (in Japanese)

[Ish1] H.Ishii, S.Shiode and T.Nishida, "Stochastic spanning tree problem", *Discreate Applied Mathematics*, Vol. 3 (1981) 263-273.

[Ish2] H.Ishii, M.Tada and T.Nishida, "Fuzzy transpotation problem", *J. of Japan Society for Fuzzy Theory and Systems*, Vol.2, No.1 (1990) 79-84. (in Japanese)

[Ish3] H.Ishii, M.Tada and T.Nishida, "Two scheduling problems with fuzzy duedates", *Fuzzy Sets and Systems*, 46 (1992) 339-347.

[Ish4] H. Ishii, "Fuzzy scheduling", *Textbook on fuzzy OR*, (Japan Society for Fuzzy Theory and Systems, 1992). (in Japanese)

[Ish5] H.Ishii and M.Tada, "Single machine scheduling problem with fuzzy precedence relations", *European Journal of Operational Research*, (to appear).

[Kau] A.Kaufmann and M.M.Gupta, *Introduction to fuzzy arithmetic / Theory and applications*, (Van Nostrand Reinhold, New York, 1991).

[Kuh] H.W.Kuhn, "The hungarian method for the assignment problem", *Naval Research Logistic Quarterly*, 2 (1955) 83-97.

[Law1] E.L.Lawler, J.K.Lenstra and A.H.G.Rinooy Kan, "Minimizing maximum lateness in a two machine open shop", *Math. Oper. Res.*, 6 (1981) 153-158.

[Law2] E.L.Lawler and J.M.Moore, "A functional equation and its application to resorce allocation and sequencing problems", *Managemnt Science* 16 (1969) 77-84.

[Mas] T.Masuda, H.Ishii and T.Nishida, "Two machine scheduling problem for jobs with generalized duedates", *Math. Japonica*, 30 (1985) 117-125.

[Meg1] N.Megiddo, "Combinatorial optimization with rational objective functions", *Math. Oper. Res.*, 4 (1979) 414-424.

[Meg2] N.Megiddo, "Linear time algorithm for linear programming in R^3 and related problems", *SIAM J. Comput.*, 12 (1983) 759-776.

[Moh] R.H.Mohring, F.J.Radermacher and G.Weiss, "Stochastic scheduling problems I: general strategies", *Zeitschrift fur Operations Research*, 28 (1984) 193-260.

[Mun] J. Munkres, "Algorithms for the assignment and transportation problems", *SIAM. J.*, 10 (1962) 196-210.

[Oku] T. Okuda, "Statistical inference from fuzzy data", *J. of Japan Society for Fuzzy Theory and Systems*, Vol.4, No.1 (1992) 41-51. (in Japanese)

[Orl] E.L. Hannan, "On formalization of a general fuzzy mathematical programming problem", *Fuzzy Sets and Systems*, Vol.3, (1980) 311-321.

[Sak1] M. Sakawa and H. Yano, "An interactive fuzzy decision making method using constraint problems", *IEEE Transaction on Systems, Man, and Cybernetics*, Vol.SMC-16, (1986) 179-182.

[Sak2] M. Sakawa, "Fuzzy interactive multiobjective programming", *J. of Japan Society for Fuzzy Theory and Systems*, Vol.4, No.1 (1992) 11-20. (in Japanese)

[Smi] W. E. Smith, "Various optimizers for sigle stage production", *Nav. Res. Log. Quart.*, Vol. 3 (1956) 59-66.

[Tad1] M.Tada, H.Ishii and T.Nishida, "A Generalized version of one machine mean flow time minimization problem", *Technology Reports of the Osaka University*, Vol.36, No.1819 (1986) 1-4.

[Tad2] M.Tada, H.Ishii and T.Nishida, "A Generalized version of one machine maximum lateness problem", *J. of the O.R. society of Japan*, Vol.32, No.3 (1989) 383-389.

[Tad3] M.Tada, H.Ishii and T.Nishida, "Weighted fuzzy due date scheduling problem", *Asian-Pacific Operatioms Research: APORS '88*, (1990) 415-417, (ed. by B.-H.Ahn, Elsevier Science Publishers B.V., North-Holland).

[Tad4] M.Tada, H.Ishii, T.Nishida and T.Masuda, "Fuzzy sharing problem", *Fuzzy Sets and Systems*, 33 (1989) 303-313.

[Tad5] M.Tada, H.Ishii and T.Nishida, "Fuzzy transportation problem with integral flow", *Math. Japonica*, 35, No.2 (1990) 335-341.

[Tad6] M.Tada, "One machine scheduling problem with fuzzy processing times", *J. of Business Studies Ryukoku University*, Vol.33, No.3 (1993) 1-11.

[Tad7] M.Tada and H.Ishii, "Two assignment problems with degrees of satisfaction", *T. IEE Japan*, Vol.114-C, No.4 (1994) 426-429. (in Japanese)

[Tan1] H.Tanaka, T.Okuda and K.Asai "On fuzzy mathematical programming", *J. Cybernetics*, Vol.3 (1974) 37-46.

[Tan2] H.Tanaka, H.Ichihashi and K.Asai, "A formulation of fuzzy linear programming problem based on comparison of fuzzy numbers", *Control and Cybernetics*, Vol.13 (1984) 185-194.

[Wat] J. Watada, "Methods for fuzzy Classification", *J. of Japan Society for Fuzzy Theory and Systems*, Vol.4, No.1 (1992) 61-73. (in Japanese)

[Zad] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 8 (1965) 94-102.

[Zim1] H. J. Zimmermann, "Description and optimization of fuzzy systems", *Int. Journal of General Systems*, 2 (1976) 209-215.

[Zim2] H. J. Zimmermann, "Fuzzy programming and linear programming with several objective functions", *Fuzzy Sets and Systems*, 1 (1978) 44-55.

[Zim3] H. J. Zimmermann, *Fuzzy set theory and its applications*, (Kluwer, Boston, 2nd revised ed. 1991).